In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler,OneHotEncoder,LabelEncoder,StandardS
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,co
from sklearn.model_selection import KFold,cross_validate
```

Read data

In [2]:
```
col_names = ['fLength', 'fWidth', 'fSize', 'fConc', 'fConc1', 'fAsym', 'fM3Long', '
df = pd.read_csv("magic04.data")
df.columns = col_names
df.head()
```

Out[2]:

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	f
0	31.6036	11.7235	2.5185	0.5303	0.3773	26.2722	23.8238	-9.9574	6.3609	205.
1	162.0520	136.0310	4.0612	0.0374	0.0187	116.7410	-64.8580	-45.2160	76.9600	256.
2	23.8172	9.5728	2.3385	0.6147	0.3922	27.2107	-6.4633	-7.1513	10.4490	116.
3	75.1362	30.9205	3.1611	0.3168	0.1832	-5.5277	28.5525	21.8393	4.6480	356.
4	51.6240	21.1502	2.9085	0.2420	0.1340	50.8761	43.1887	9.8145	3.6130	238.

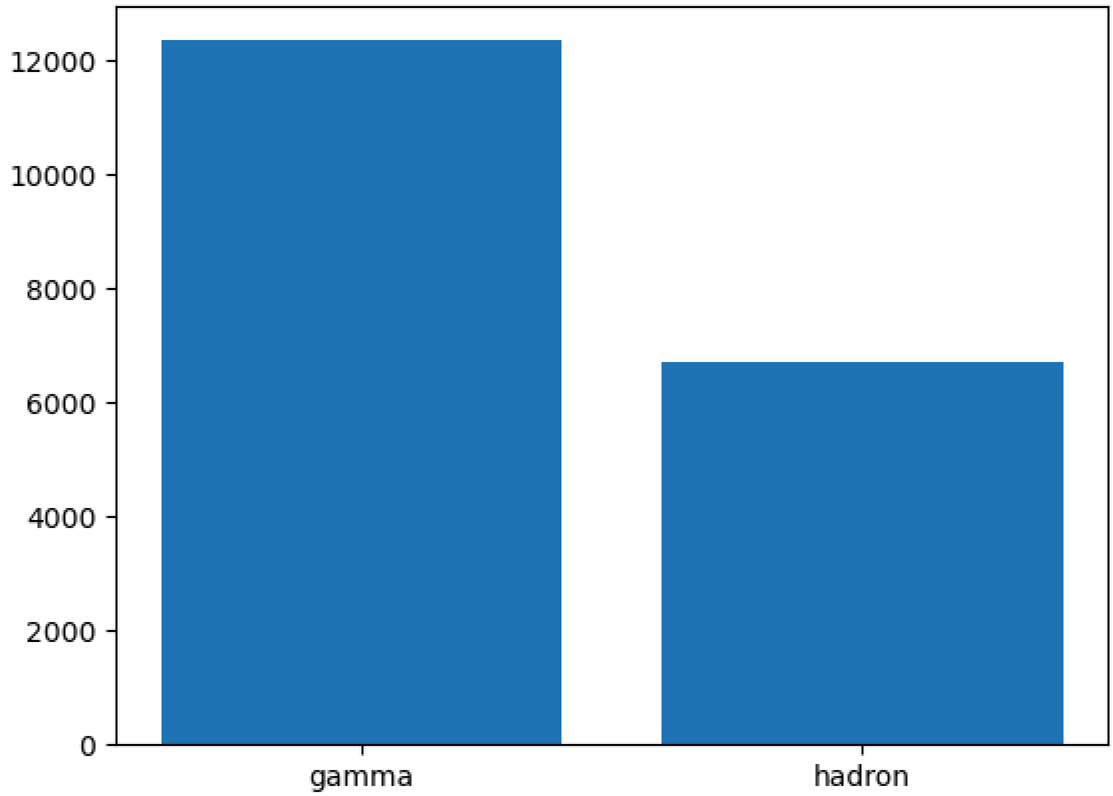
In [3]:
```
print(set(df['class']))
```

```
{'g', 'h'}
```

In [4]:
```
df.dropna(inplace=True)
```

In [5]:
```
plt.bar(['gamma','hadron'],height=[len(df[df['class']=='g']),len(df[df['class']=='h
```

Out[5]:
```
<BarContainer object of 2 artists>
```

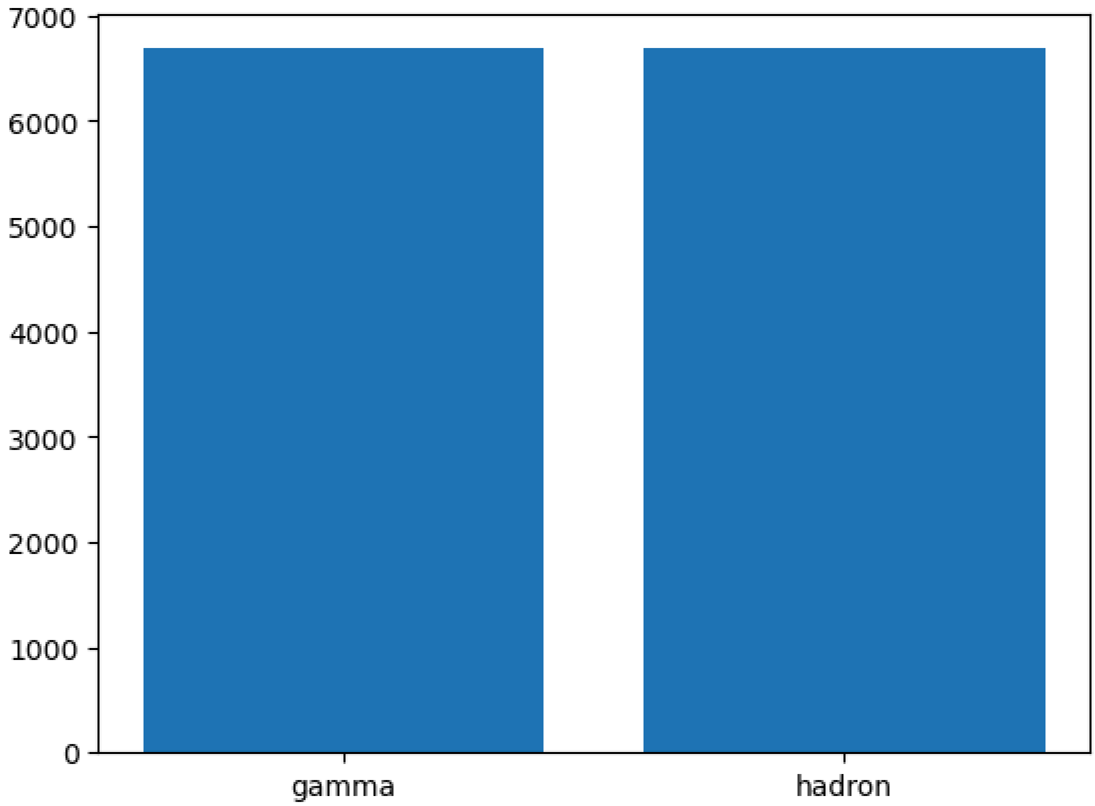


Balancing Data

In [6]:
```
df = df.groupby('class').sample(len(df[df['class']=='h']))
```

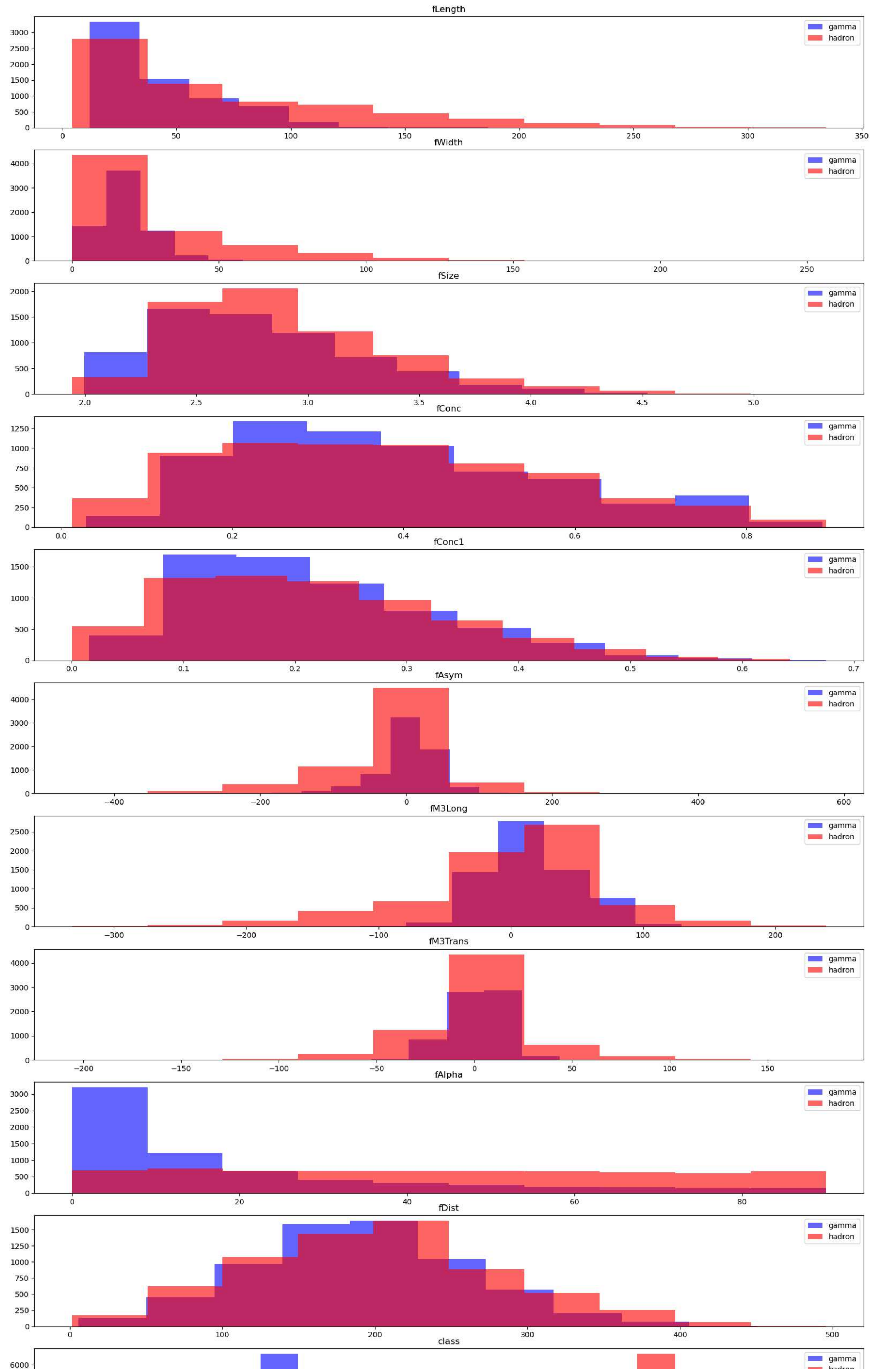
In [7]:
```
plt.bar(['gamma','hadron'],height=[len(df[df['class']=='g']),len(df[df['class']=='h
```

Out[7]: <BarContainer object of 2 artists>



Visualizing Data Columns

In [8]:
```
fig,axs = plt.subplots(11,figsize=(20,35))
for i,label in enumerate(df):
    axs[i].hist(df[df['class']=='g'][label],color='blue',alpha=0.6,label="gamma")
    axs[i].hist(df[df['class']=='h'][label],color='red',alpha=0.6,label="hadron")
    axs[i].title.set_text(label)
    axs[i].legend()
plt.show()
```


fLength
gamma
hadron
fWidth
gamma
hadron
fSize
gamma
hadron
fConc
gamma
hadron
fConc1
gamma
hadron
fAsym
gamma
hadron
fM3Long
gamma
hadron
fM3Trans
gamma
hadron
fAlpha
gamma
hadron
fDist
gamma
hadron
class
gamma

Encoding

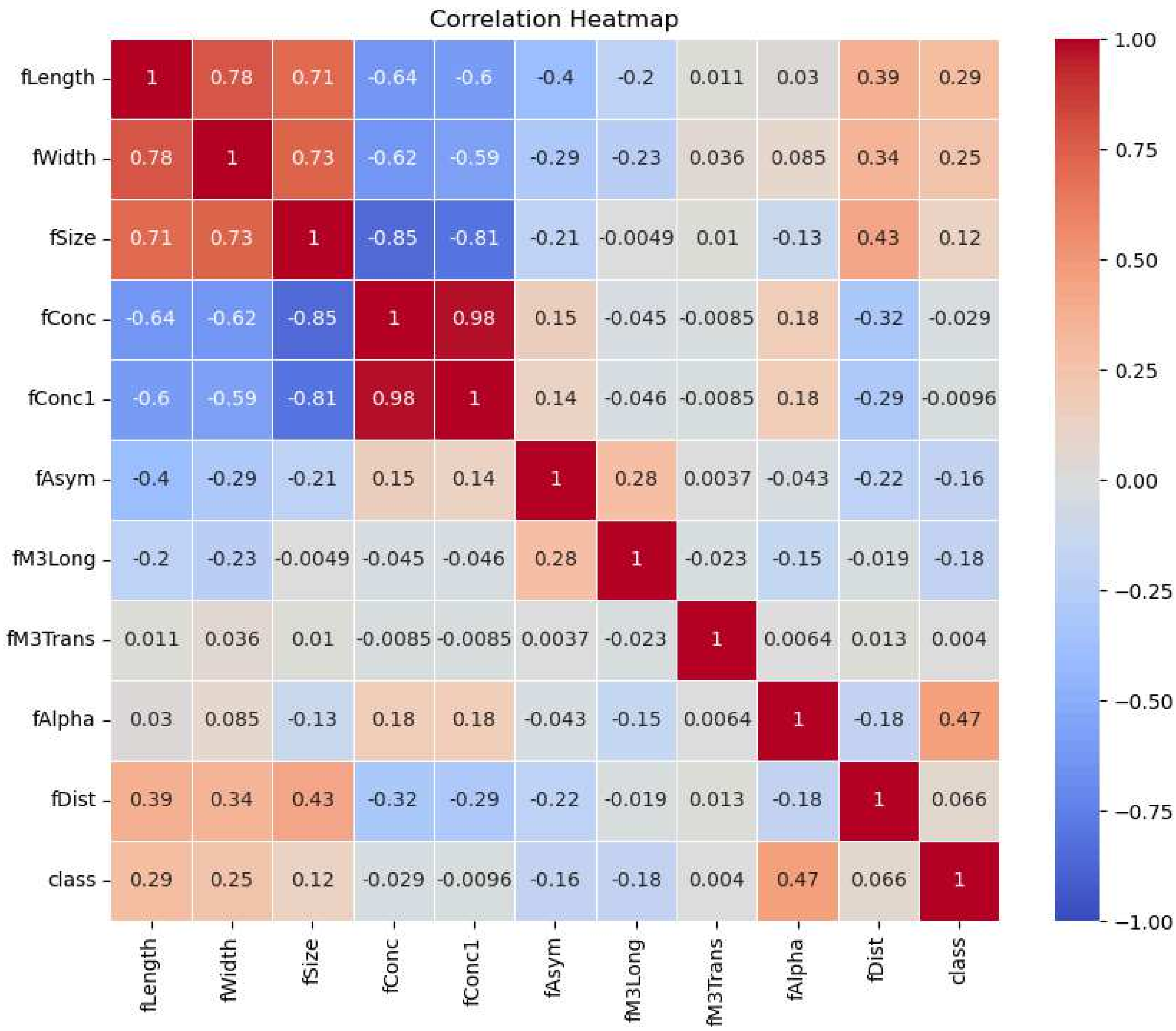
In [9]:
```
en = LabelEncoder()
df['class'] = en.fit_transform(df['class'])
```

In [10]:
```
df.head()
```

Out[10]:

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	
4710	41.7854	17.7204	2.6365	0.3210	0.1721	-17.1632	-37.2658	13.6465	0.5310	1
10250	41.8464	19.4302	3.1166	0.2217	0.1235	11.5249	27.9829	-11.9406	0.0920	1
3165	95.2562	27.9901	3.1992	0.2851	0.1489	-66.8185	-71.2218	-24.3355	3.6810	3
2855	13.0339	11.1611	2.0810	0.7552	0.3942	14.9112	-4.3689	11.7397	86.3258	
4526	48.2713	11.7368	2.5192	0.5446	0.2950	12.0288	40.4022	1.4286	10.3490	2

In [11]:
```
import seaborn as sns
correlation_matrix = df.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", vmin=-1, vmax=1, linew

plt.title("Correlation Heatmap")

plt.show()
```



	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist	class
fLength	1	0.78	0.71	-0.64	-0.6	-0.4	-0.2	0.011	0.03	0.39	0.29
fWidth	0.78	1	0.73	-0.62	-0.59	-0.29	-0.23	0.036	0.085	0.34	0.25
fSize	0.71	0.73	1	-0.85	-0.81	-0.21	-0.0049	0.01	-0.13	0.43	0.12
fConc	-0.64	-0.62	-0.85	1	0.98	0.15	-0.045	-0.0085	0.18	-0.32	-0.029
fConc1	-0.6	-0.59	-0.81	0.98	1	0.14	-0.046	-0.0085	0.18	-0.29	-0.0096
fAsym	-0.4	-0.29	-0.21	0.15	0.14	1	0.28	0.0037	-0.043	-0.22	-0.16
fM3Long	-0.2	-0.23	-0.0049	-0.045	-0.046	0.28	1	-0.023	-0.15	-0.019	-0.18
fM3Trans	0.011	0.036	0.01	-0.0085	-0.0085	0.0037	-0.023	1	0.0064	0.013	0.004
fAlpha	0.03	0.085	-0.13	0.18	0.18	-0.043	-0.15	0.0064	1	-0.18	0.47
fDist	0.39	0.34	0.43	-0.32	-0.29	-0.22	-0.019	0.013	-0.18	1	0.066
class	0.29	0.25	0.12	-0.029	-0.0096	-0.16	-0.18	0.004	0.47	0.066	1

splitting

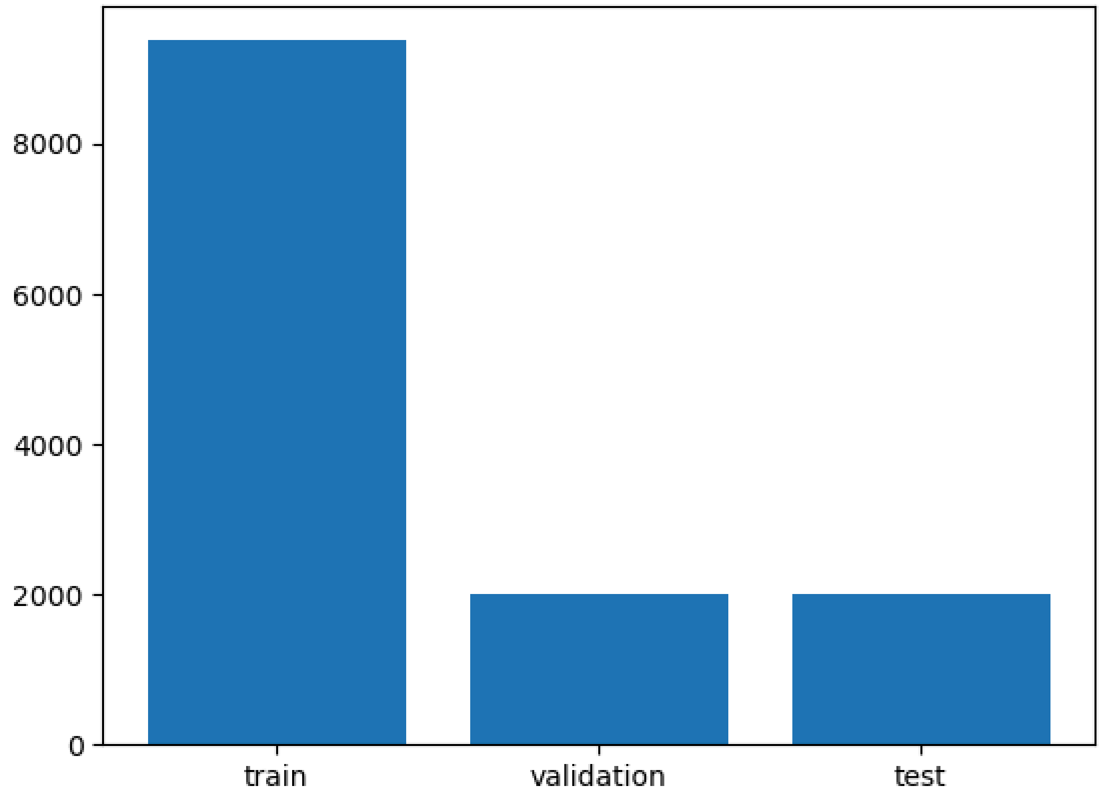
In [12]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [13]:
```
norm = StandardScaler()
X = norm.fit_transform(X)
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=42,shuffle=Tru
X_val, X_test, y_val, y_test = train_test_split(X_test,y_test,random_state=42,shuff
```

In [15]:
```
plt.bar(['train','validation','test'],[len(y_train),len(y_val),len(y_test)])
```

Out[15]: <BarContainer object of 3 artists>



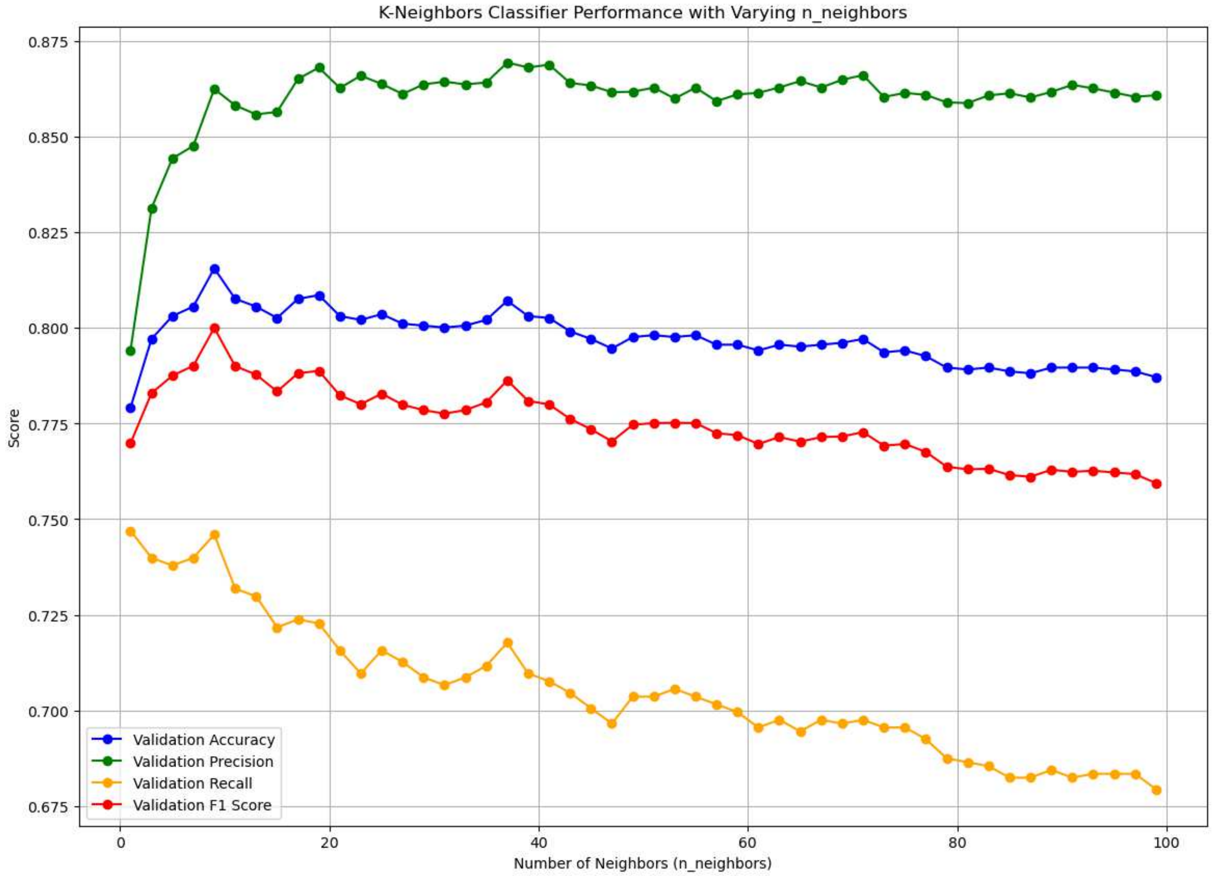
In [16]:
```
neighbors = range(1,100,2)
```

In [17]:
```
def compare(neighbors, X_train, X_val, y_train, y_val):
    accs = []
    prec = []
    rec = []
    f1_scores = []
    CMs = []

    for i in neighbors:
        model = KNeighborsClassifier(n_neighbors=i)

        # Train the model on the full training data
        model.fit(X_train, y_train)

        # Predict on training and validation sets
        y_train_pred = model.predict(X_train)
        y_val_pred = model.predict(X_val)

        # Calculate metrics for training set
        train_acc = accuracy_score(y_train, y_train_pred)
        train_precision = precision_score(y_train, y_train_pred)
        train_recall = recall_score(y_train, y_train_pred)
        train_f1 = f1_score(y_train, y_train_pred)

        # Calculate metrics for validation set
        val_acc = accuracy_score(y_val, y_val_pred)
        val_precision = precision_score(y_val, y_val_pred)
        val_recall = recall_score(y_val, y_val_pred)
        val_f1 = f1_score(y_val, y_val_pred)
```

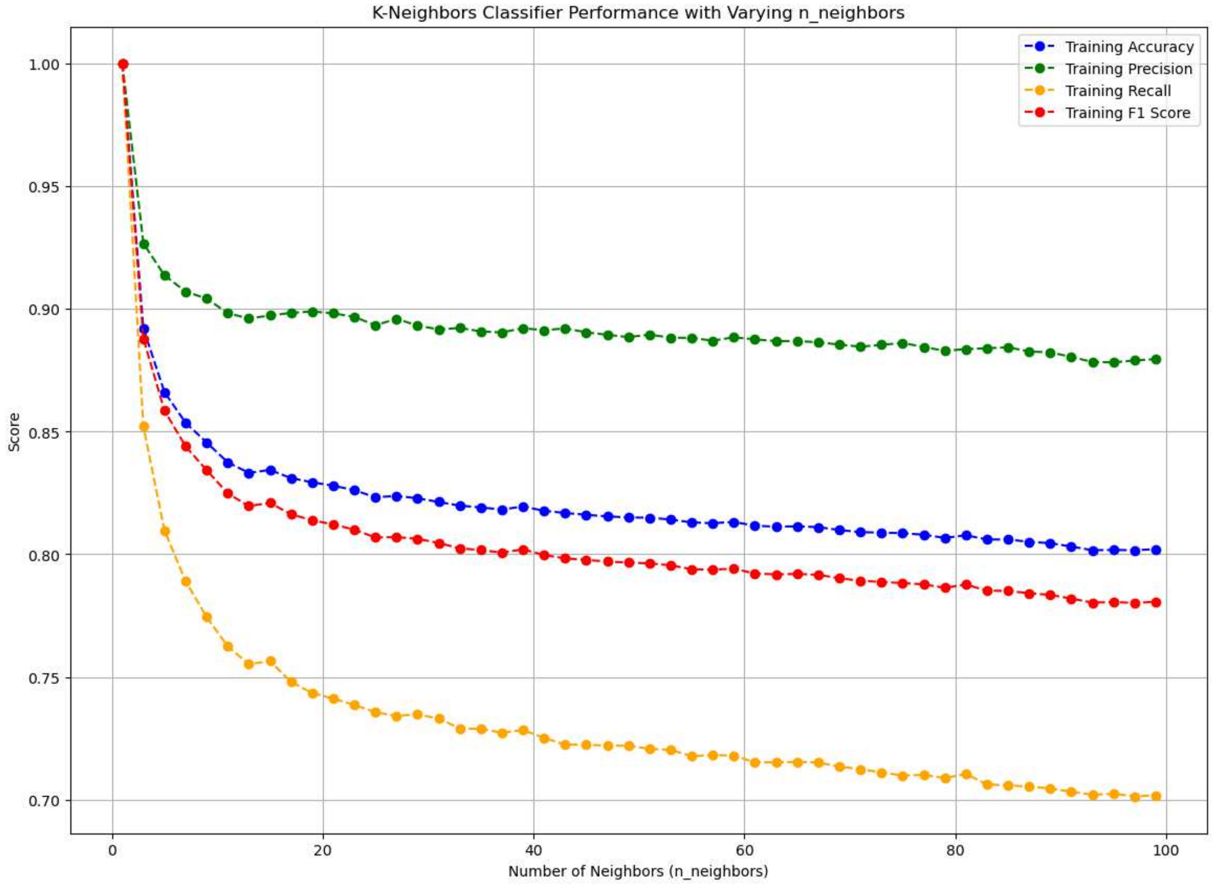
```
        # Confusion matrix for validation
        cm = confusion_matrix(y_val, y_val_pred)

        # Append training and validation metrics
        accs.append((val_acc,train_acc))
        prec.append((val_precision,train_precision))
        rec.append((val_recall,train_recall))
        f1_scores.append((val_f1,train_f1))
        CMs.append(cm)

    return accs, prec, rec, f1_scores, CMs
```

In [27]:
```
accs,prec,rec,f1_scores,CMs = compare(neighbors,X_train,X_val,y_train,y_val)
```

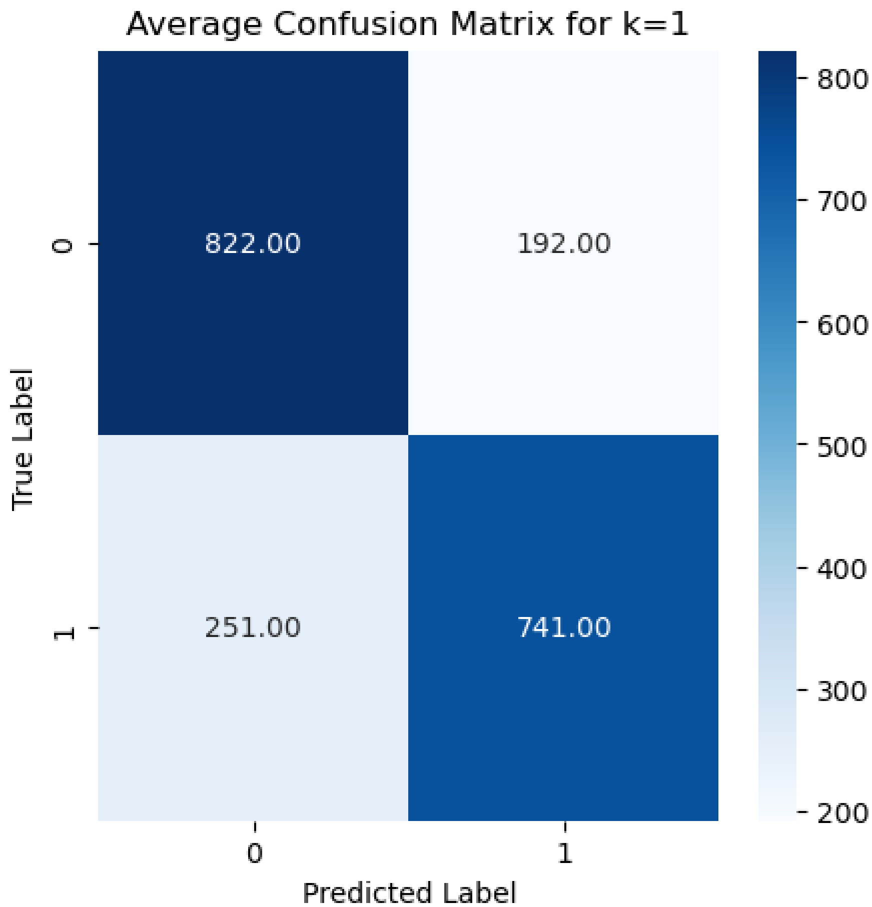
In [28]:
```
val_accs = [x[0] for x in accs]
train_accs = [x[1] for x in accs]
val_prec = [x[0] for x in prec]
train_prec = [x[1] for x in prec]
val_rec = [x[0] for x in rec]
train_rec = [x[1] for x in rec]
val_f1 = [x[0] for x in f1_scores]
train_f1 = [x[1] for x in f1_scores]

# Plot the results
plt.figure(figsize=(14, 10))

# Plot accuracy
plt.plot(neighbors, val_accs, label='Validation Accuracy', marker='o', color='blue'

# Plot precision
plt.plot(neighbors, val_prec, label='Validation Precision', marker='o', color='gree

# Plot recall
plt.plot(neighbors, val_rec, label='Validation Recall', marker='o', color='orange')

# Plot F1 score
plt.plot(neighbors, val_f1, label='Validation F1 Score', marker='o', color='red')

# Add labels, title, and legend
plt.title('K-Neighbors Classifier Performance with Varying n_neighbors')
plt.xlabel('Number of Neighbors (n_neighbors)')
plt.ylabel('Score')
plt.legend()
plt.grid(True)
plt.show()
```



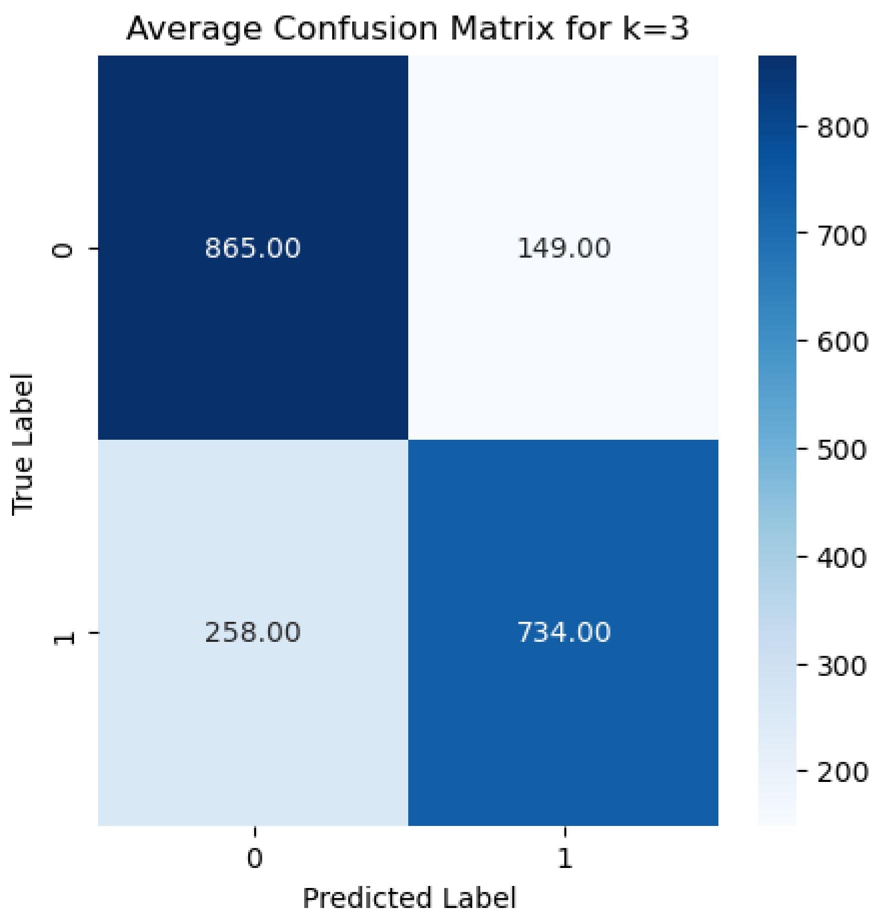
In [29]:
```
plt.figure(figsize=(14, 10))
plt.plot(neighbors, train_accs, label='Training Accuracy', marker='o', linestyle='-
plt.plot(neighbors, train_prec, label='Training Precision', marker='o', linestyle='
plt.plot(neighbors, train_rec, label='Training Recall', marker='o', linestyle='--',
plt.plot(neighbors, train_f1, label='Training F1 Score', marker='o', linestyle='--'
plt.title('K-Neighbors Classifier Performance with Varying n_neighbors')
plt.xlabel('Number of Neighbors (n_neighbors)')
plt.ylabel('Score')
plt.legend()
plt.grid(True)
plt.show()
```



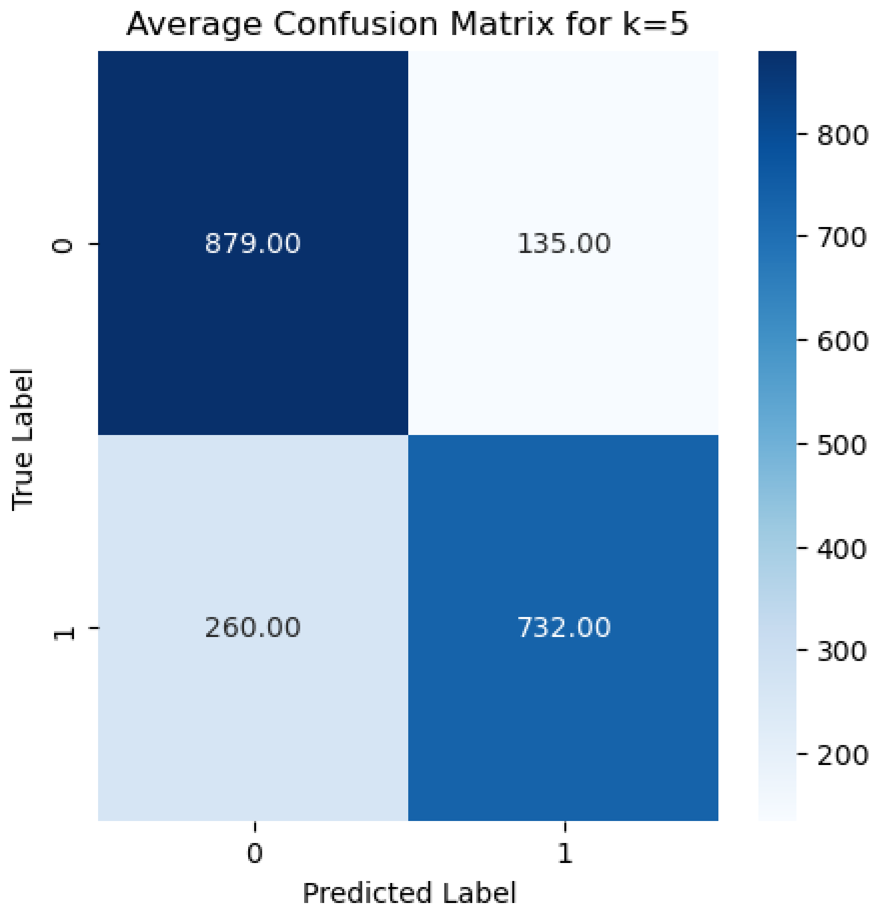
In [30]:
```python
import seaborn as sns
for i, cm in enumerate(CMs):
    plt.figure(figsize=(5, 5))
    sns.heatmap(cm, annot=True, fmt='.2f', cmap='Blues')
    plt.title(f'Average Confusion Matrix for k={neighbors[i]}')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()
```

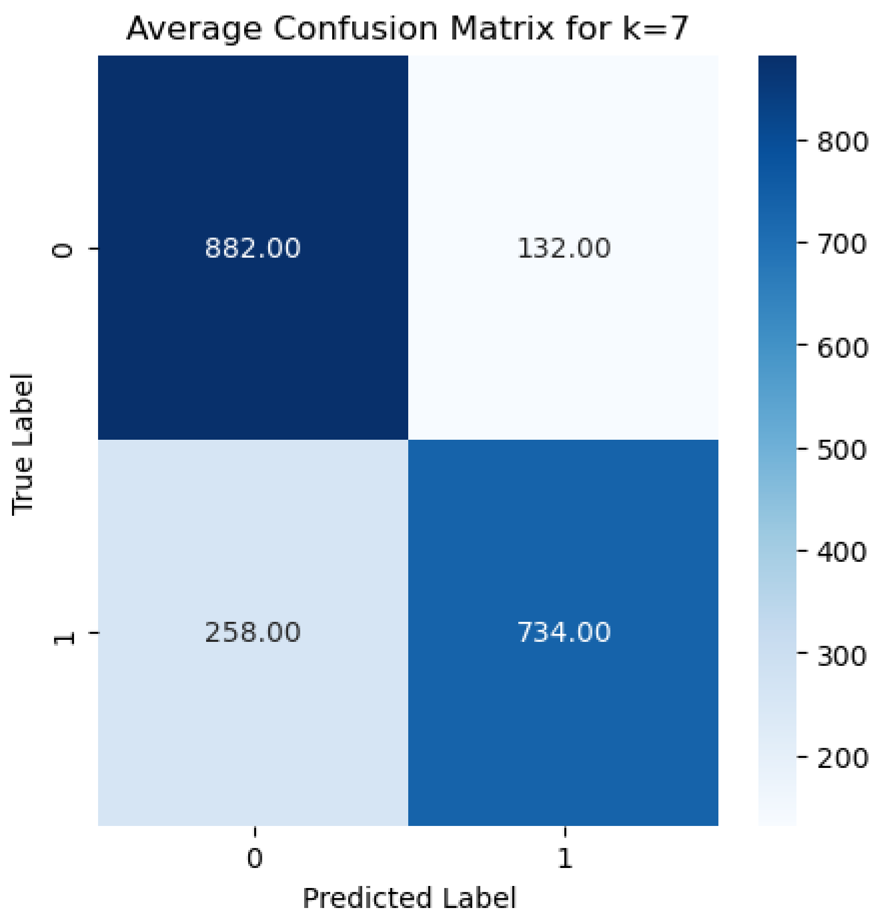


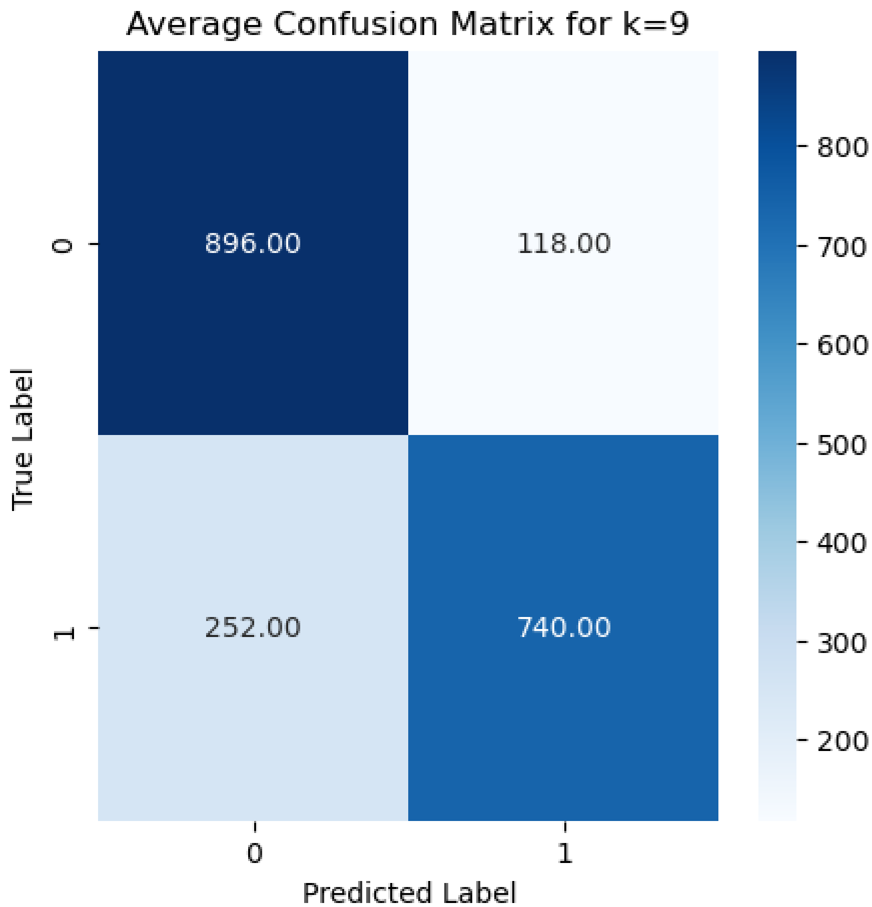
Average Confusion Matrix for k=1
822.00
192.00
251.00
741.00
True Label
Predicted Label
0
1
800
700
600
500
400
300
200

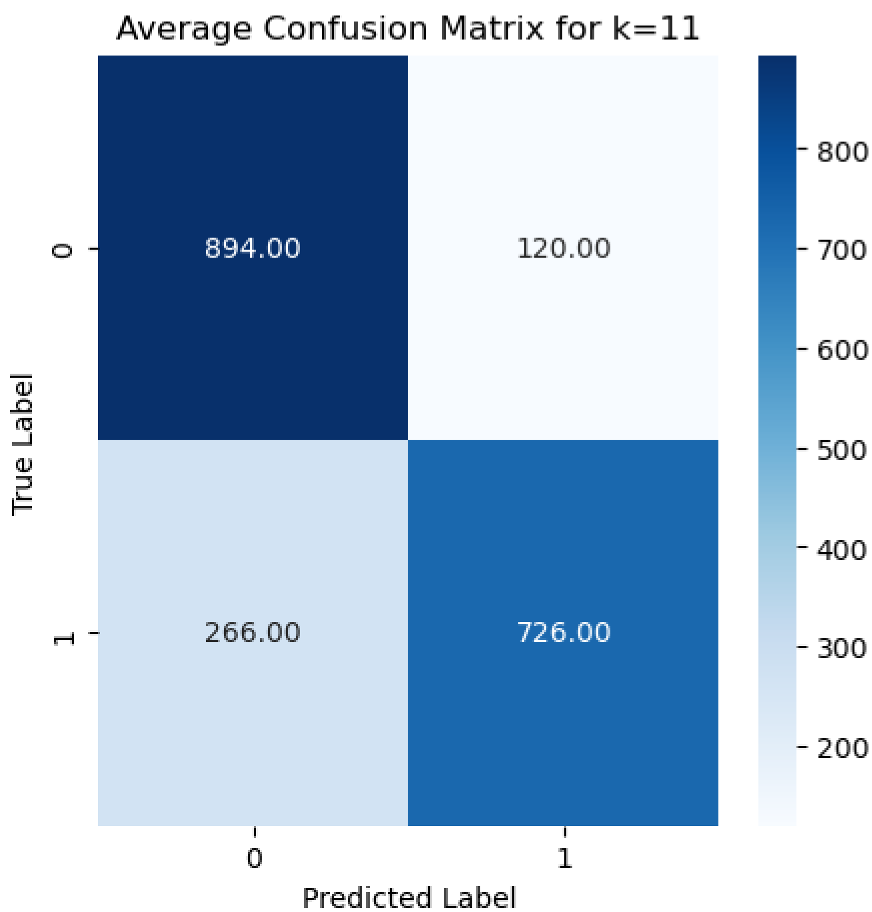


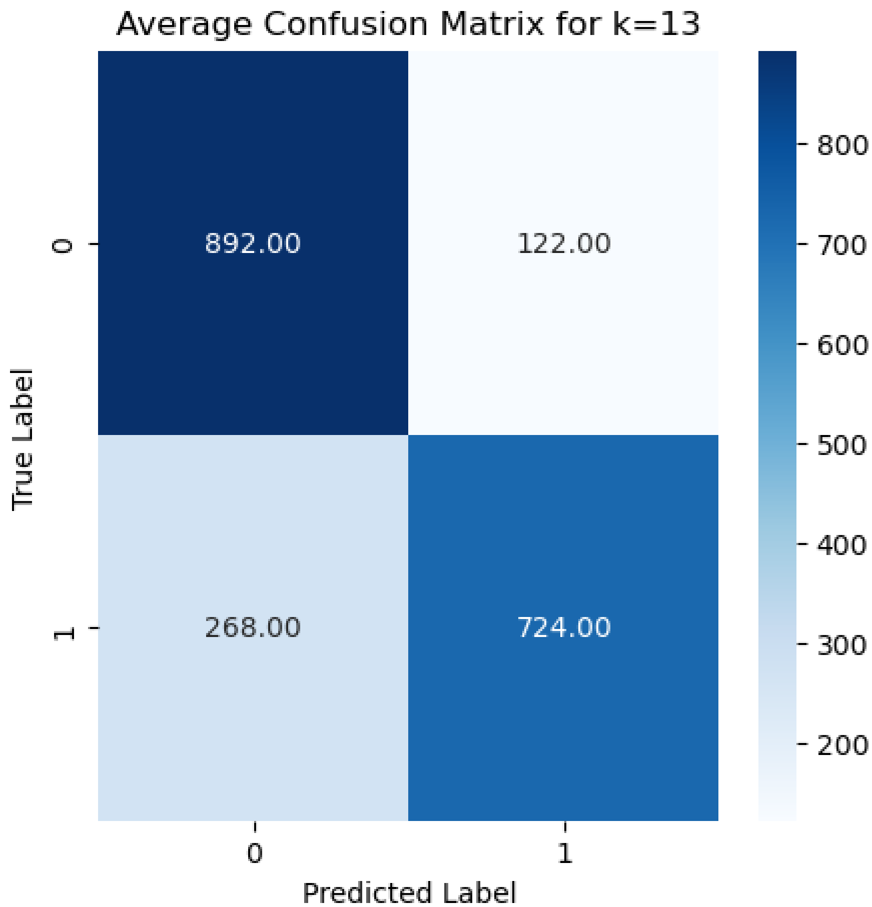
Average Confusion Matrix for k=3
865.00
149.00
258.00
734.00
True Label
Predicted Label
0
1
800
700
600
500
400
300
200

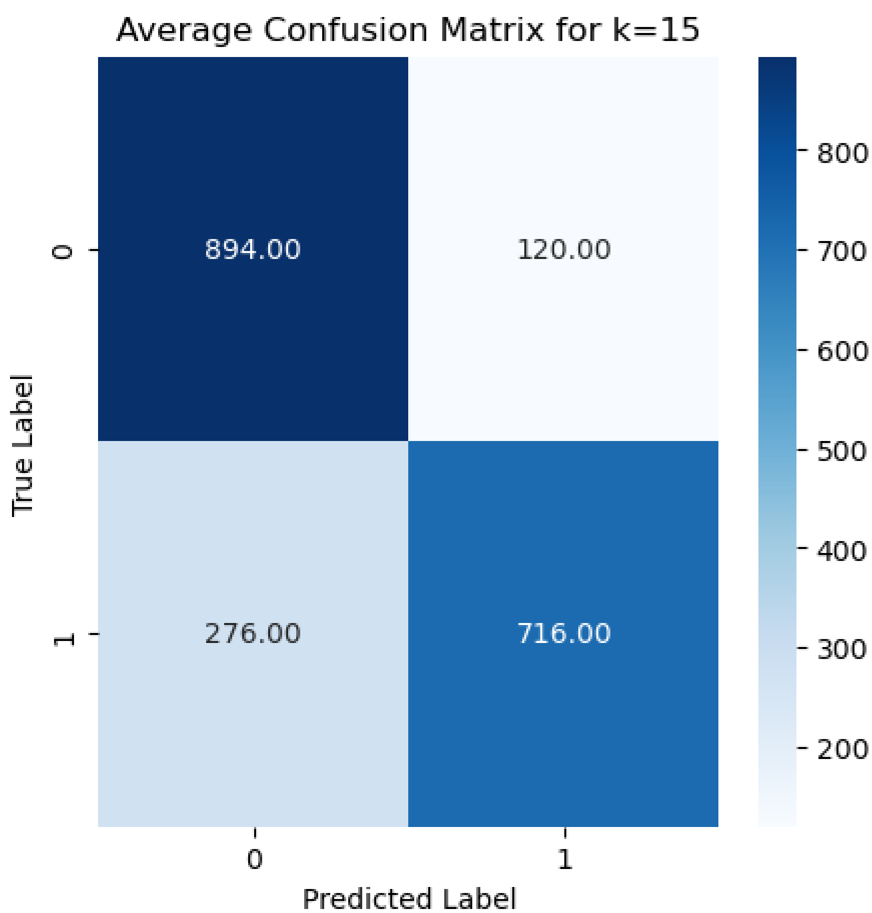

Average Confusion Matrix for k=5
True Label
Predicted Label
0
1
879.00
135.00
260.00
732.00
800
700
600
500
400
300
200

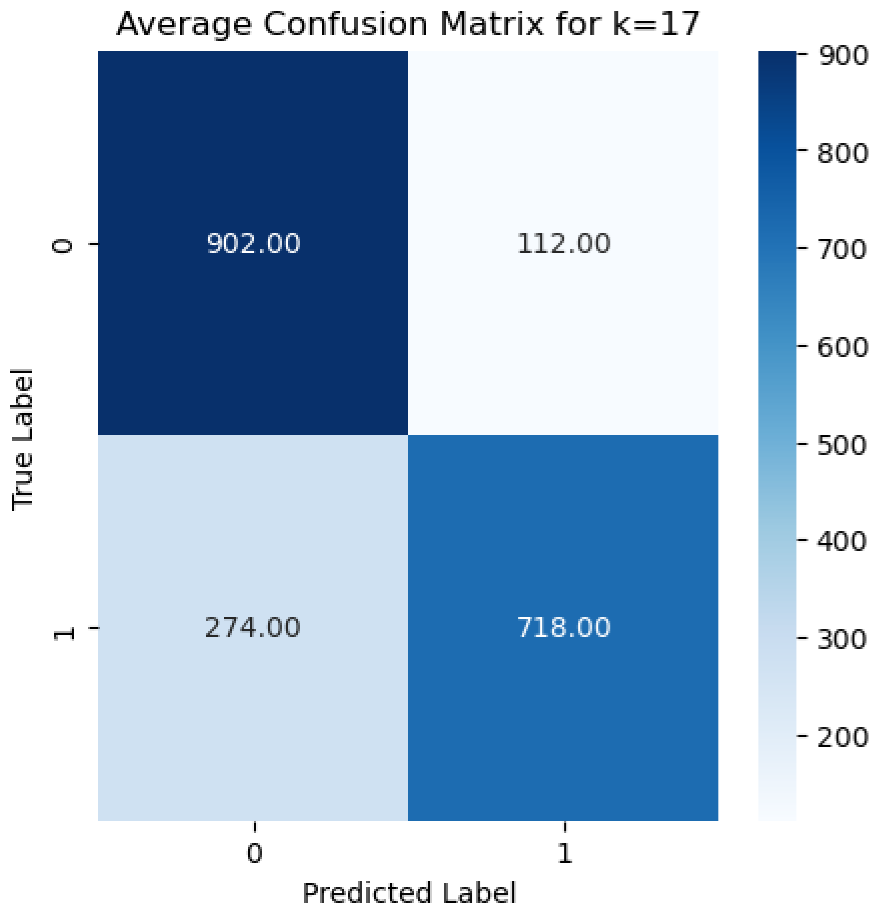

Average Confusion Matrix for k=7
True Label
Predicted Label
0
1
882.00
132.00
258.00
734.00
800
700
600
500
400
300
200

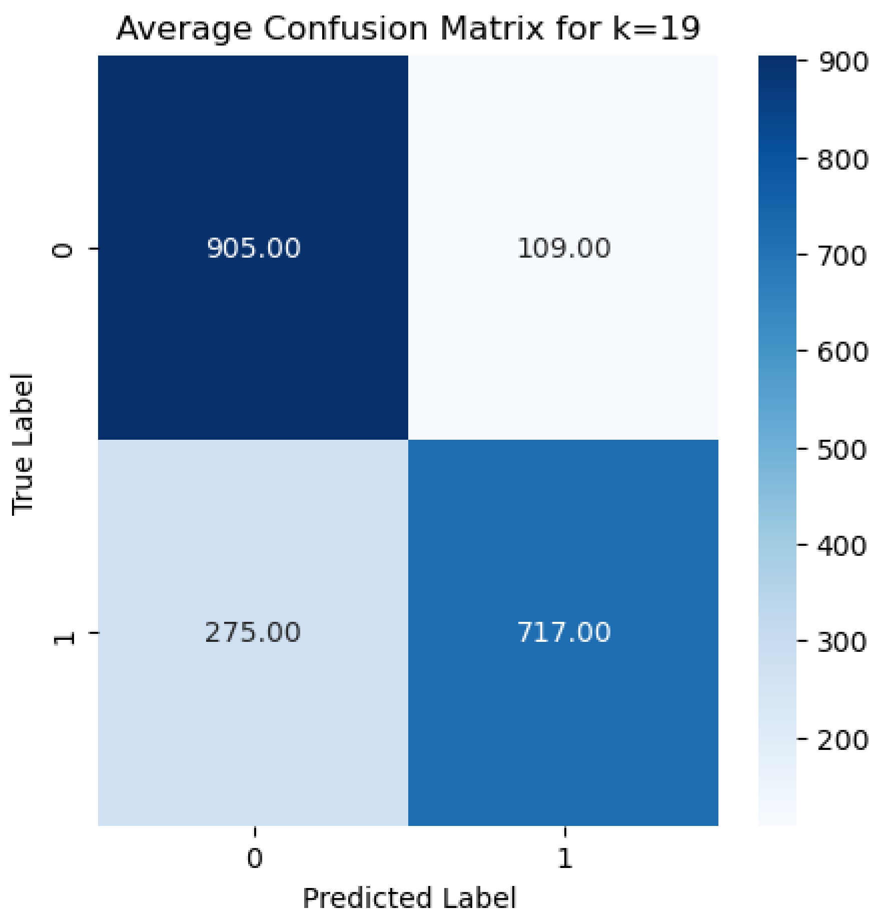

Average Confusion Matrix for k=9
True Label
Predicted Label
0
1
896.00
118.00
252.00
740.00
800
700
600
500
400
300
200

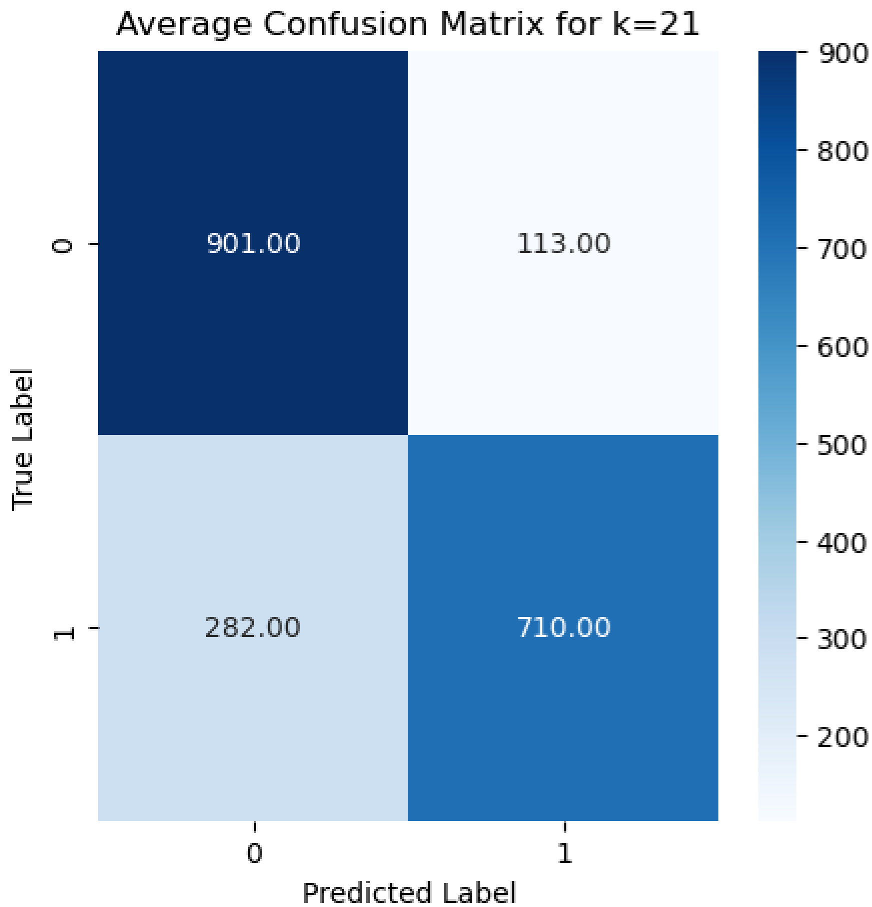

Average Confusion Matrix for k=11
True Label
Predicted Label
0
1
894.00
120.00
266.00
726.00
800
700
600
500
400
300
200

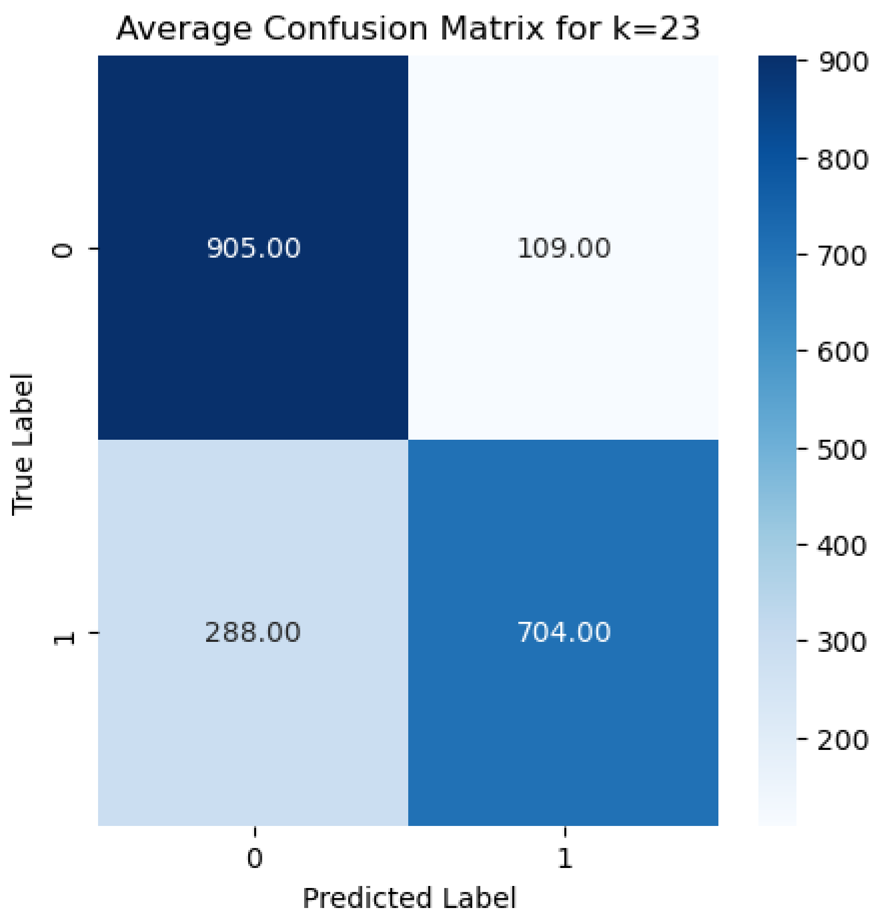

Average Confusion Matrix for k=13
892.00
122.00
268.00
724.00
True Label
Predicted Label
0
1
800
700
600
500
400
300
200

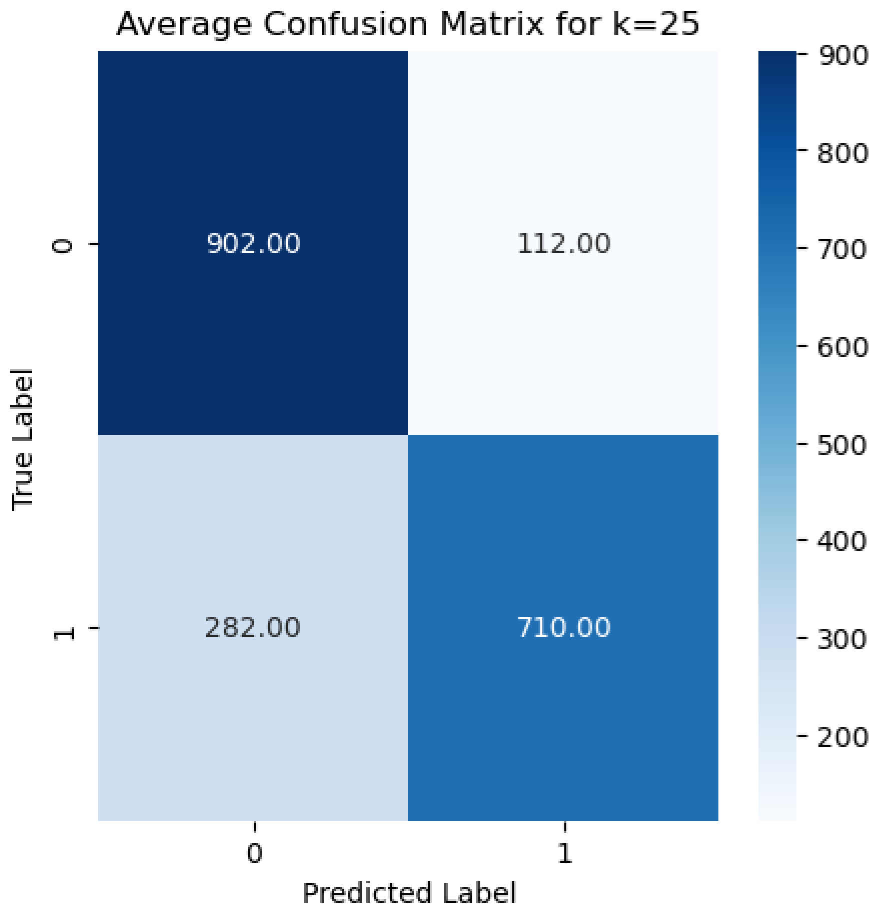

Average Confusion Matrix for k=15
894.00
120.00
276.00
716.00
True Label
Predicted Label
0
1
800
700
600
500
400
300
200

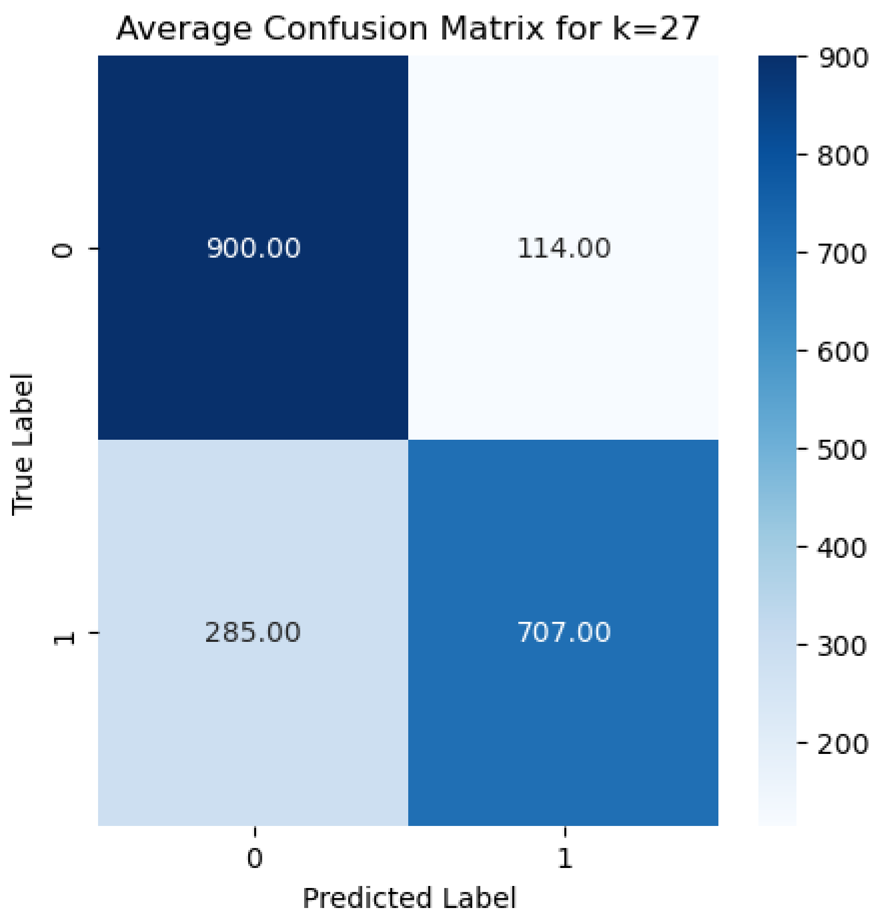

Average Confusion Matrix for k=17
902.00
112.00
274.00
718.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

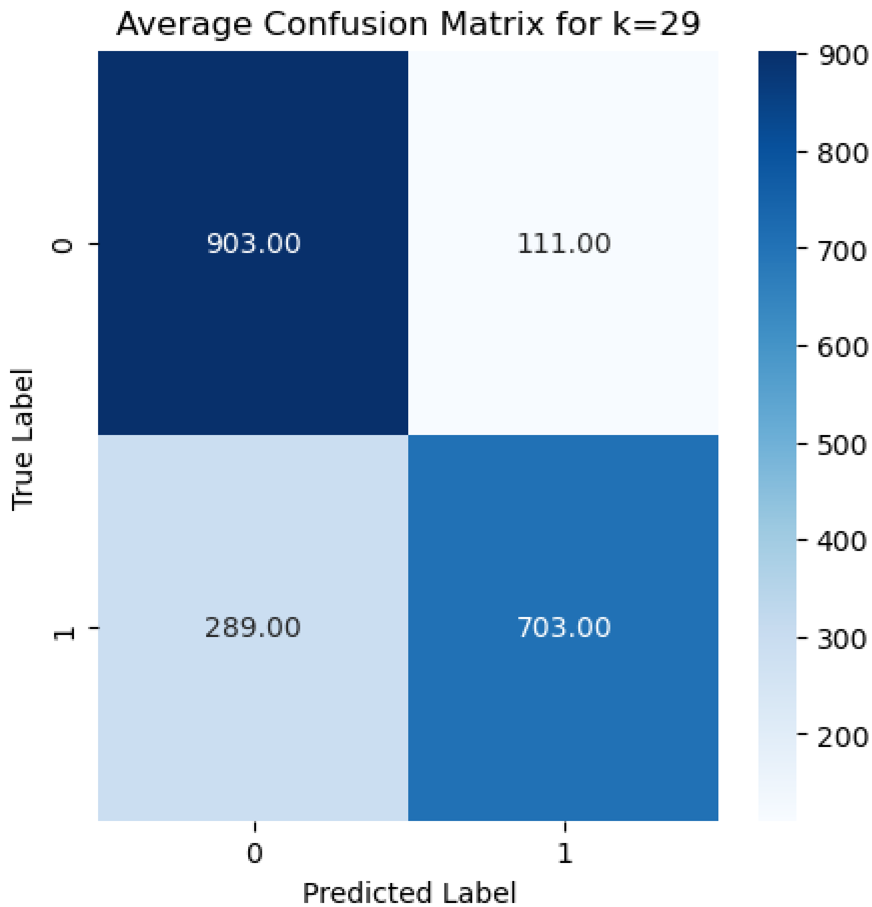

Average Confusion Matrix for k=19
905.00
109.00
275.00
717.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

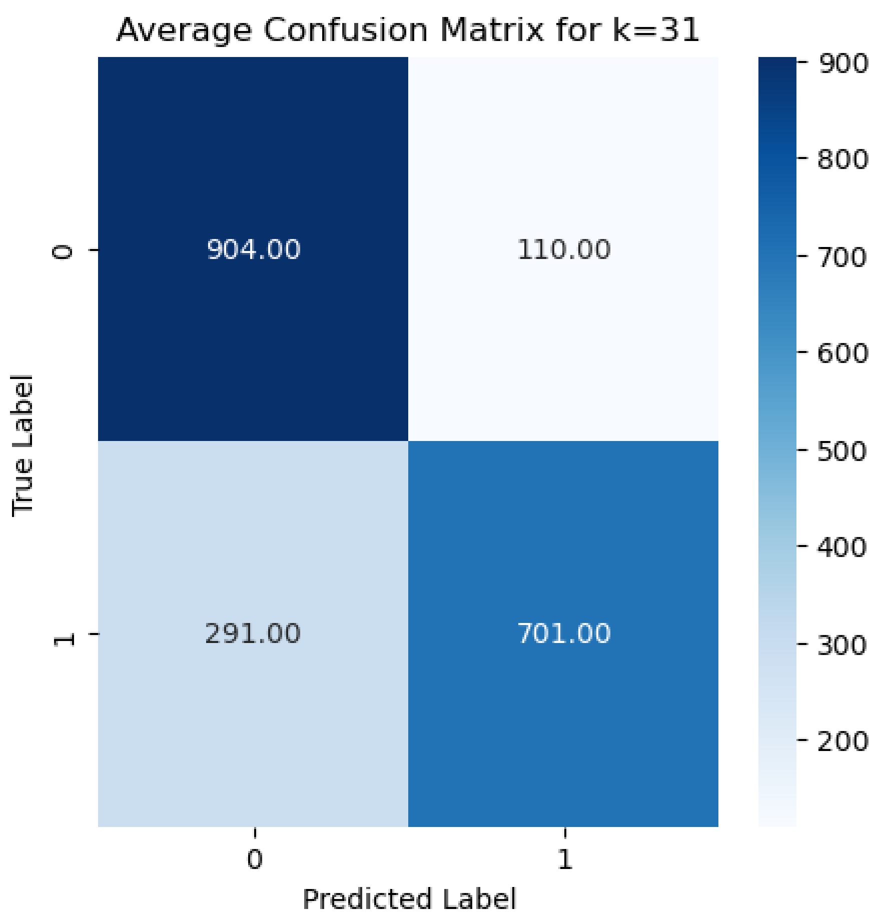

Average Confusion Matrix for k=21
True Label
Predicted Label
0
1
0
1
901.00
113.00
282.00
710.00
900
800
700
600
500
400
300
200

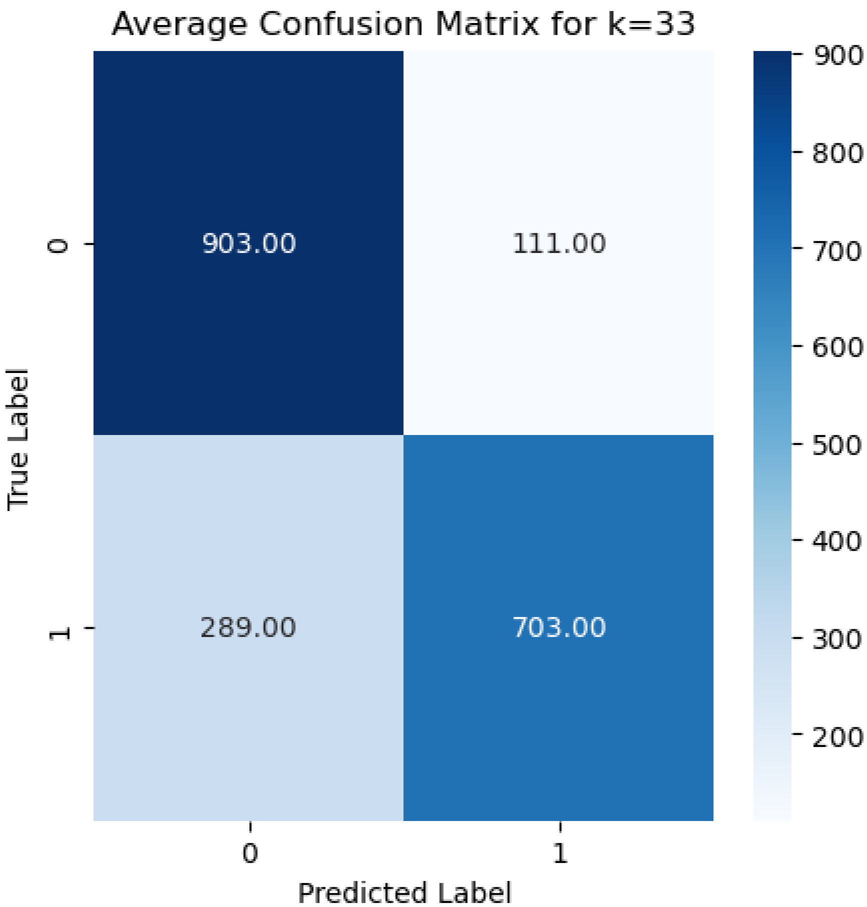

Average Confusion Matrix for k=23
True Label
Predicted Label
0
1
0
1
905.00
109.00
288.00
704.00
900
800
700
600
500
400
300
200

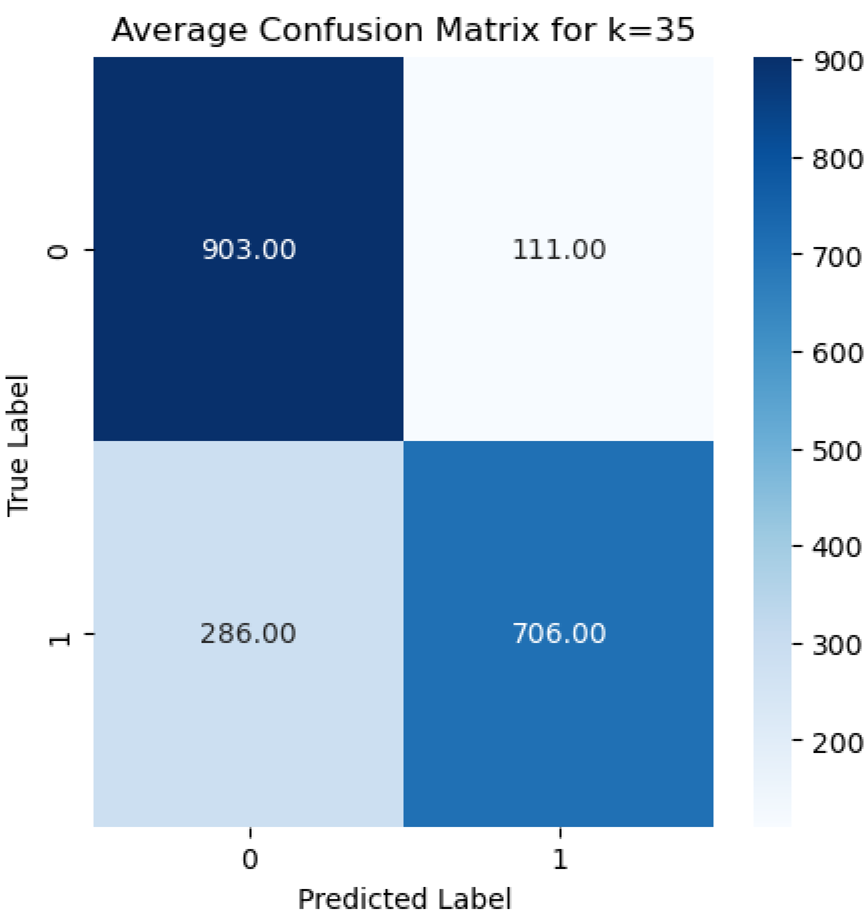

Average Confusion Matrix for k=25
True Label
Predicted Label
0
1
0
1
902.00
112.00
282.00
710.00
900
800
700
600
500
400
300
200

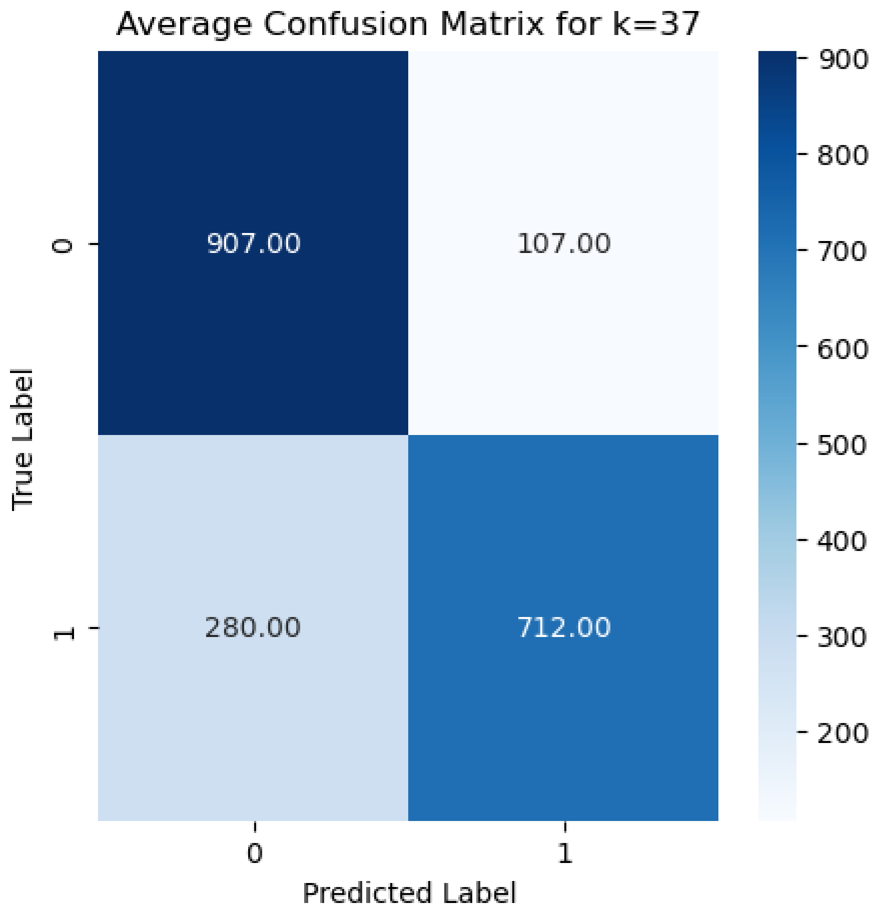

Average Confusion Matrix for k=27
True Label
Predicted Label
0
1
0
1
900.00
114.00
285.00
707.00
900
800
700
600
500
400
300
200

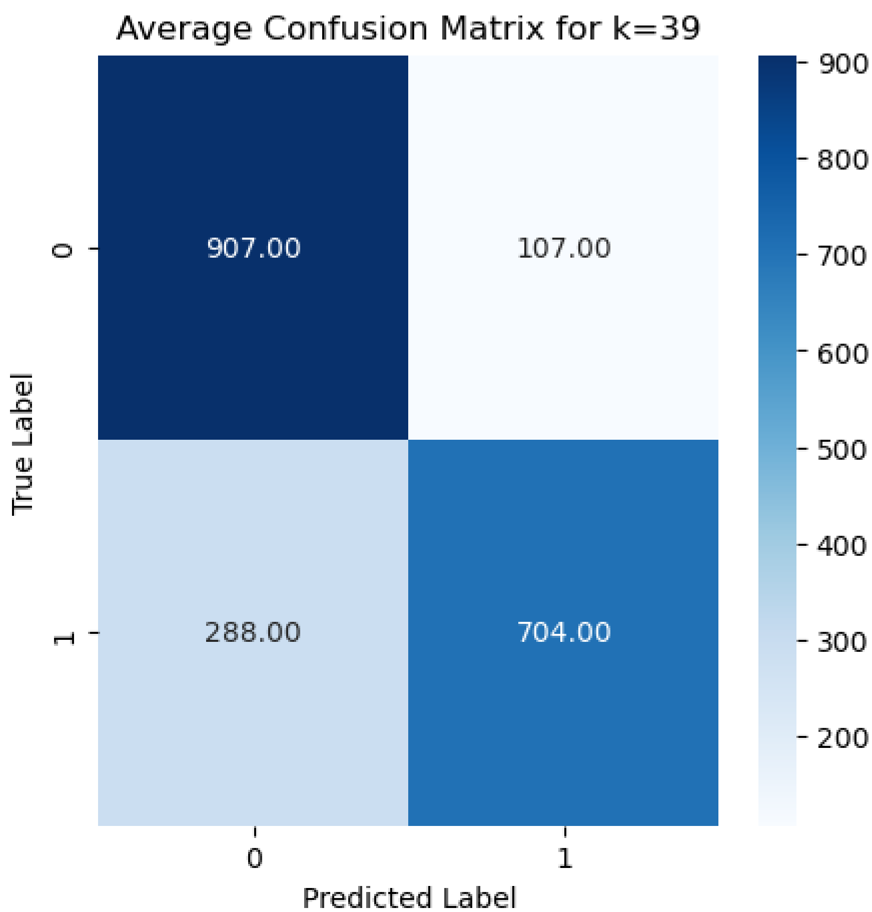

Average Confusion Matrix for k=29
903.00
111.00
289.00
703.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

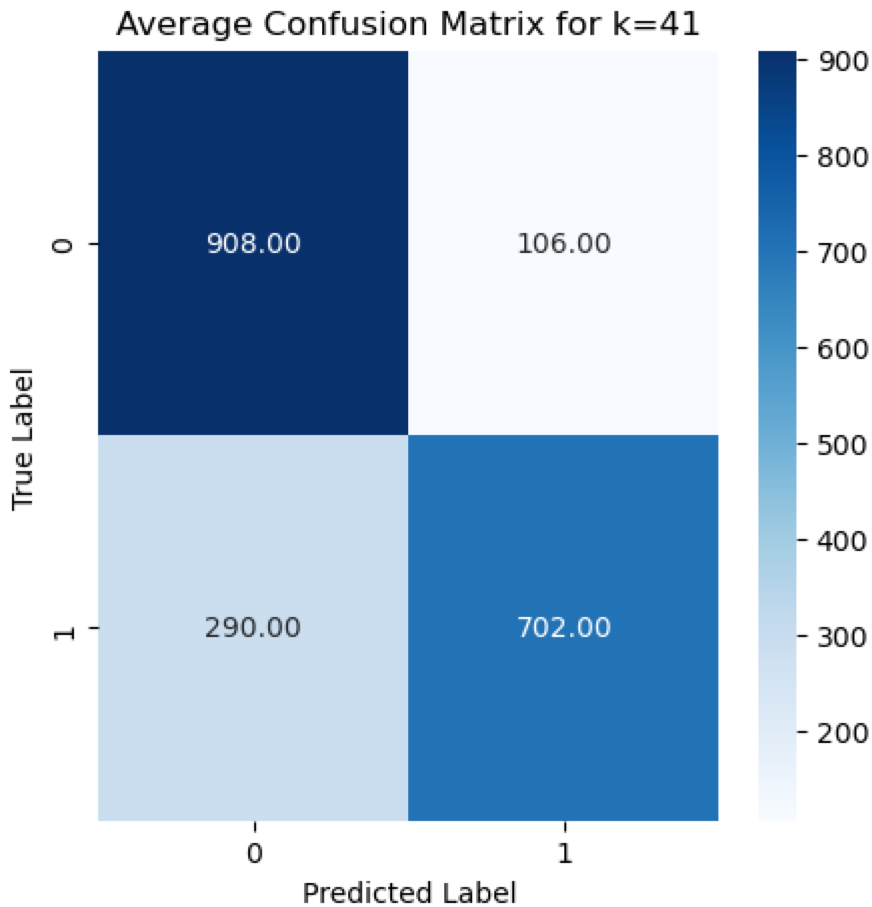

Average Confusion Matrix for k=31
904.00
110.00
291.00
701.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

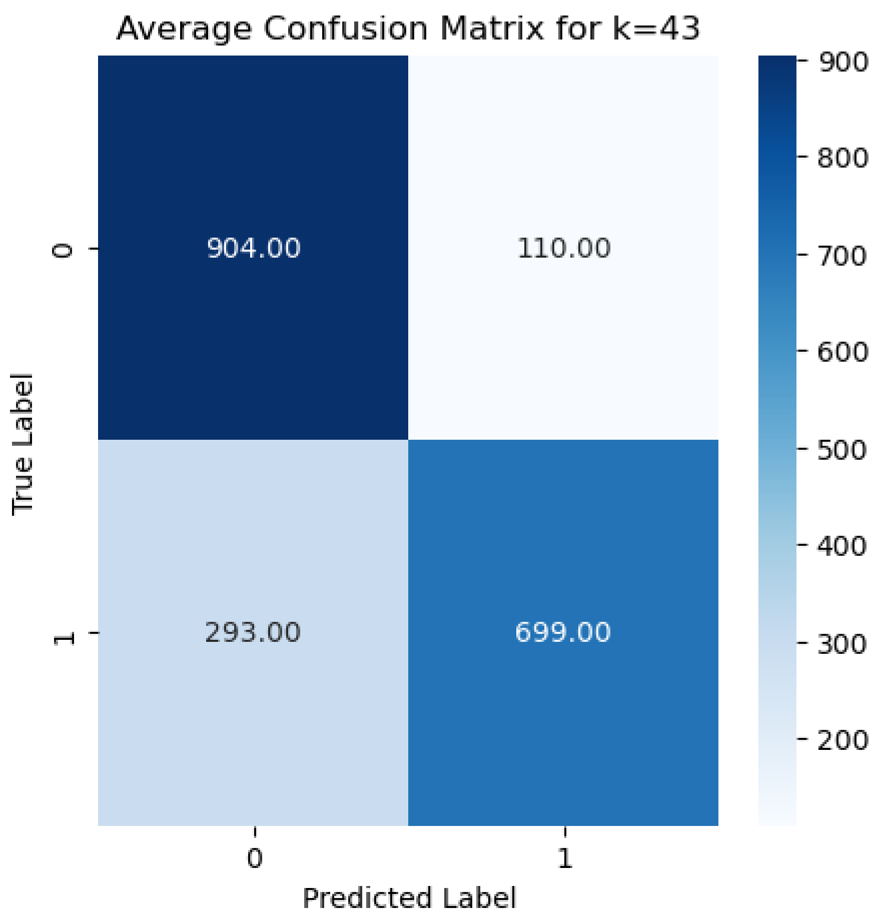

Average Confusion Matrix for k=33
903.00
111.00
289.00
703.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

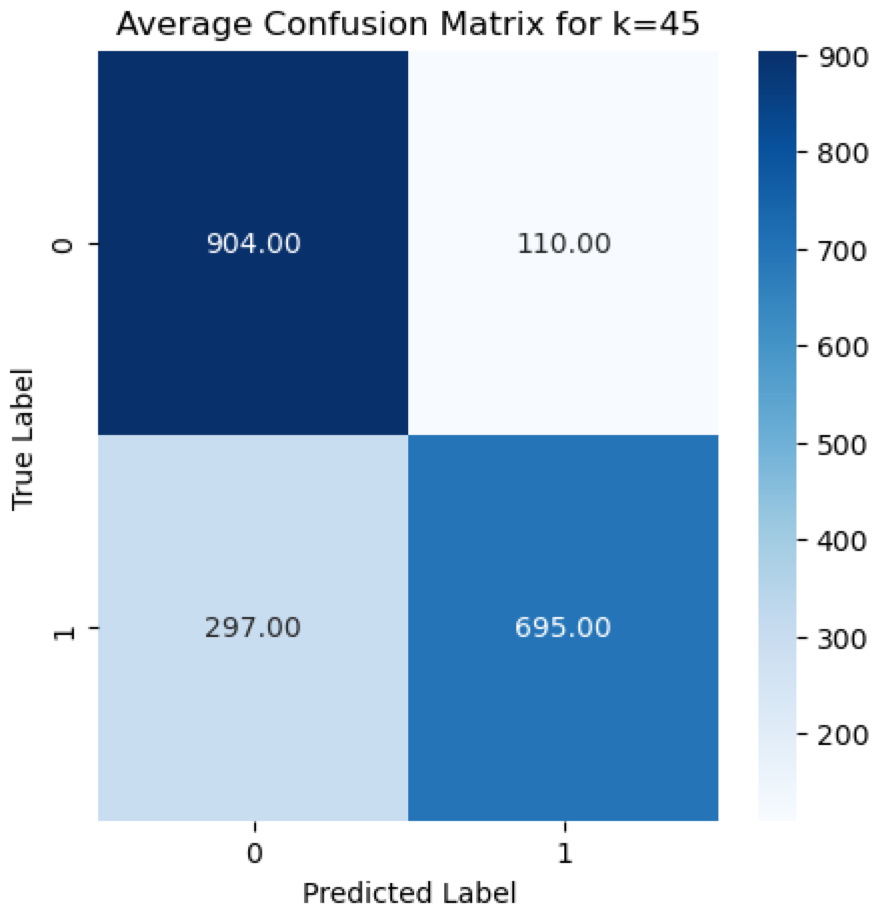

Average Confusion Matrix for k=35
903.00
111.00
286.00
706.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

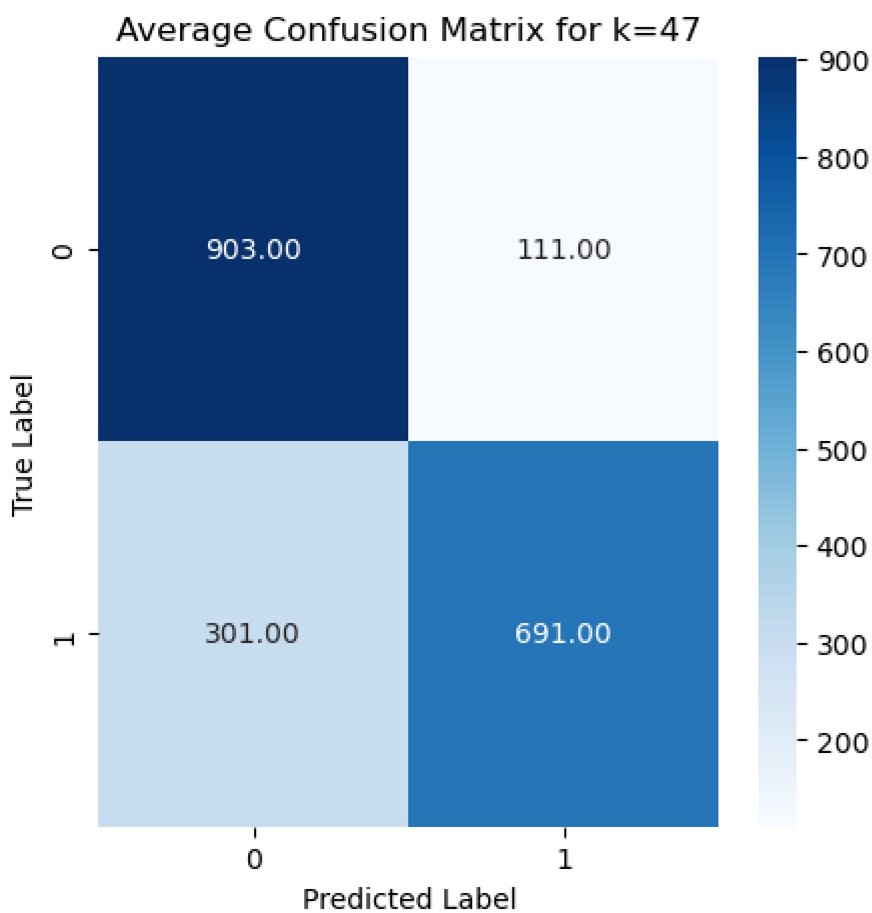

Average Confusion Matrix for k=37
907.00
107.00
280.00
712.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

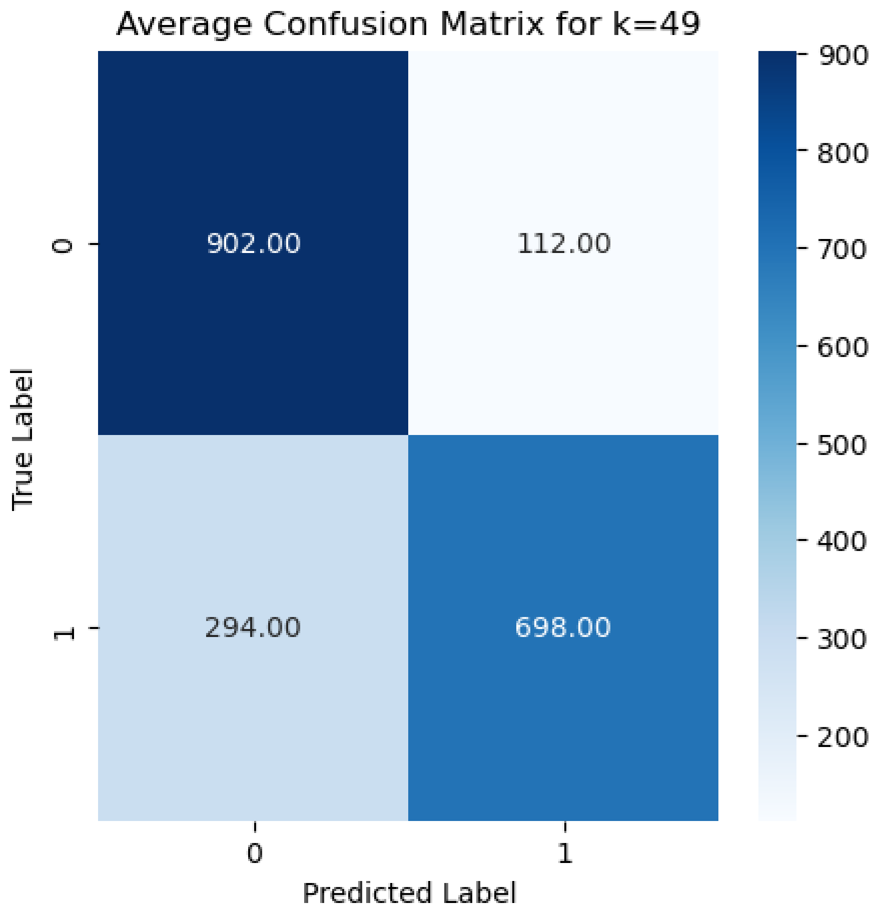

Average Confusion Matrix for k=39
907.00
107.00
288.00
704.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

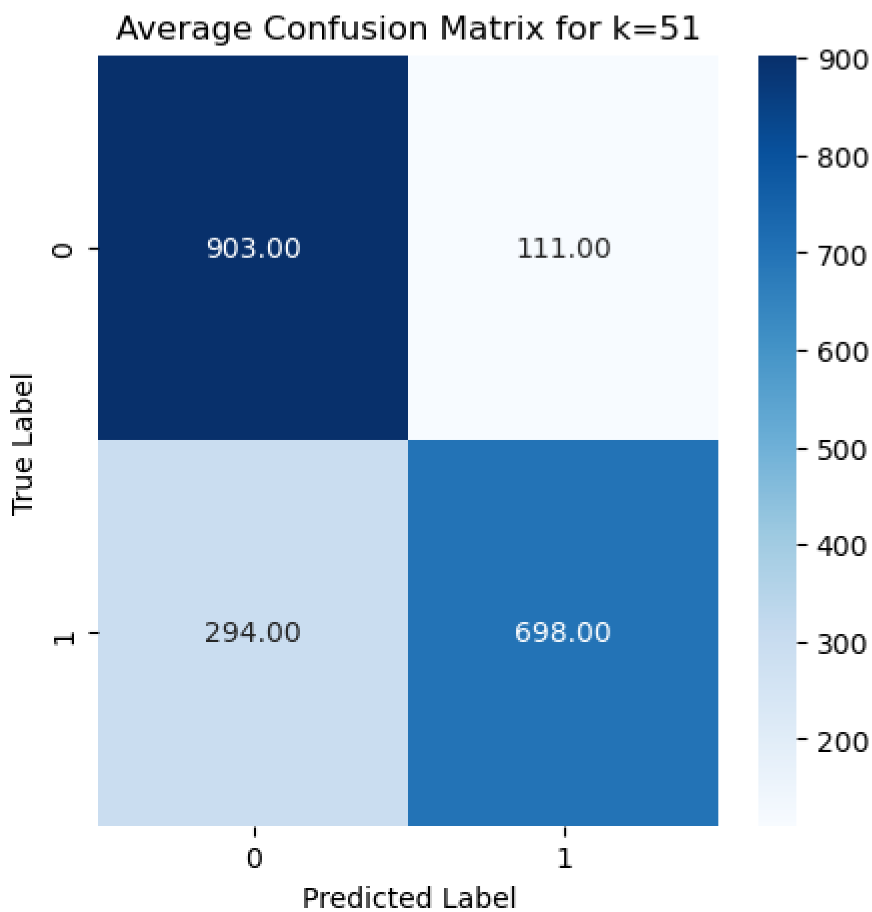

Average Confusion Matrix for k=41
908.00
106.00
290.00
702.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

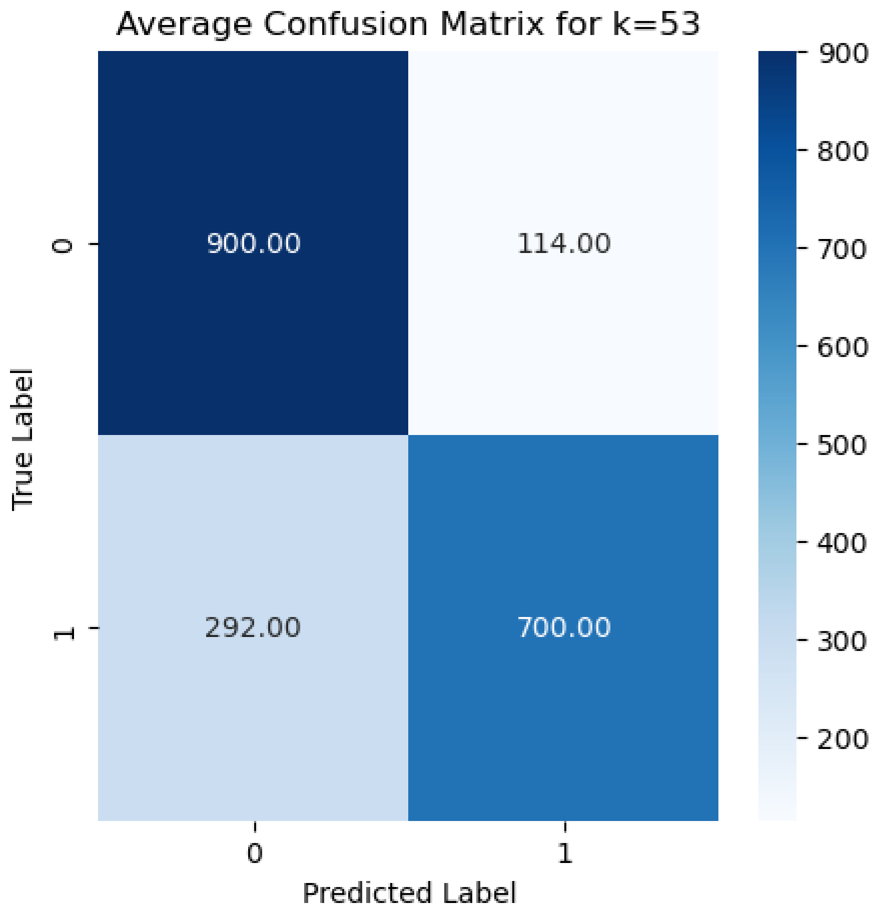

Average Confusion Matrix for k=43
904.00
110.00
293.00
699.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

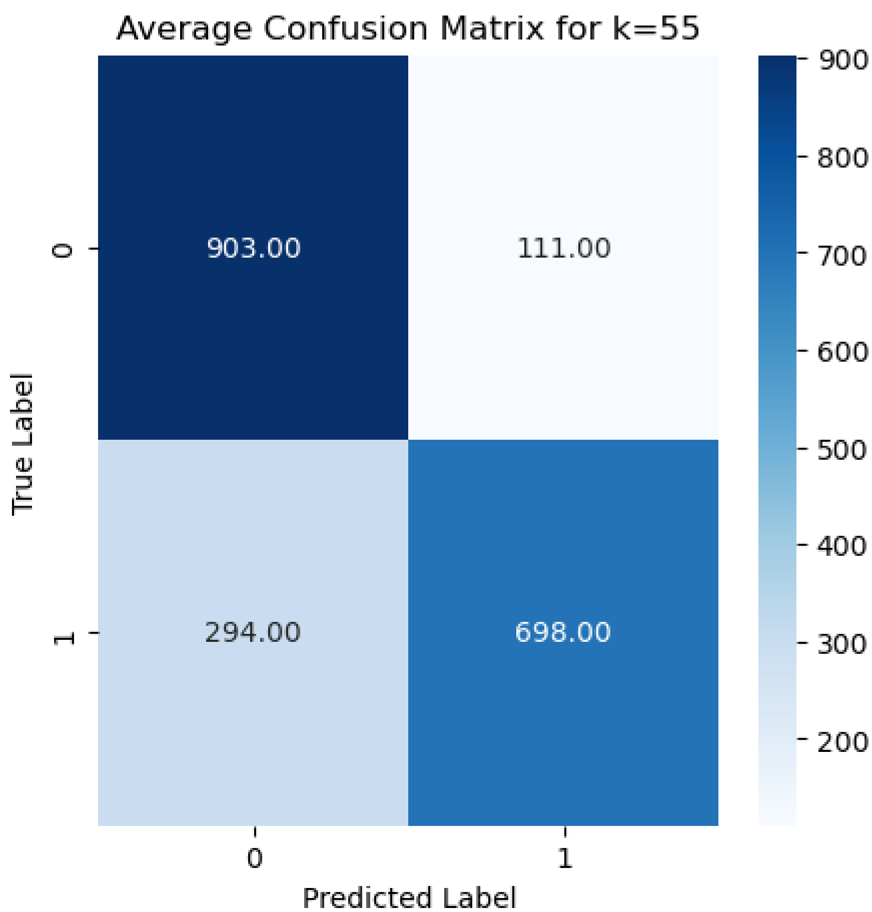

Average Confusion Matrix for k=45
904.00
110.00
297.00
695.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

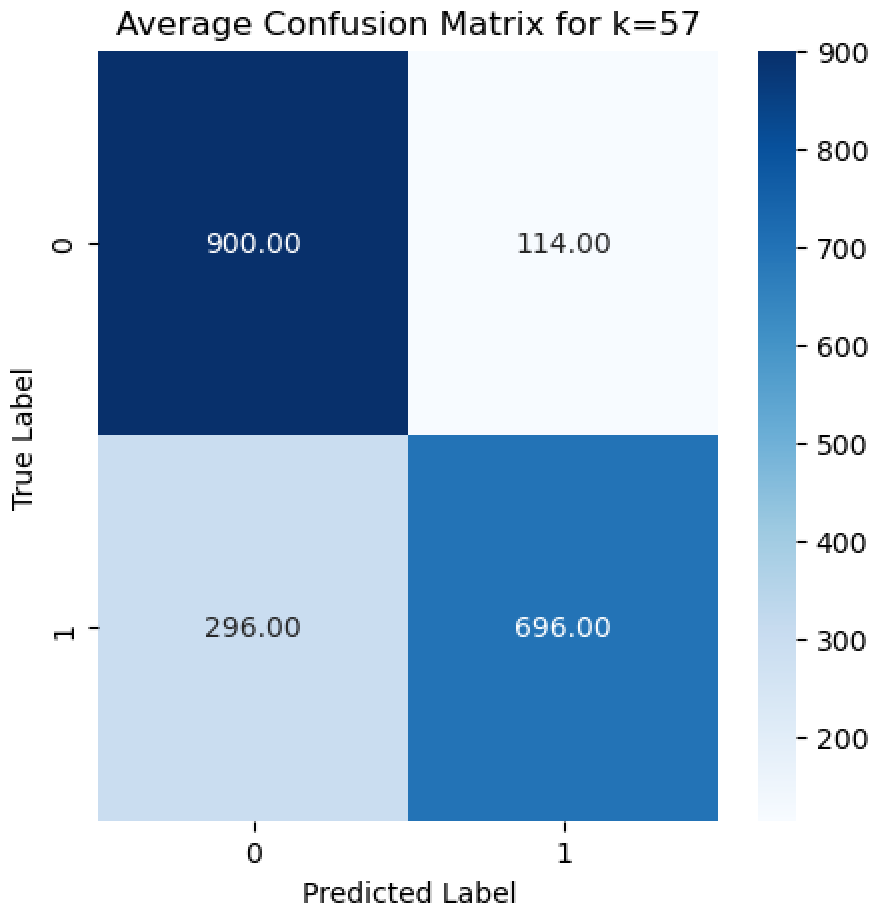

Average Confusion Matrix for k=47
903.00
111.00
301.00
691.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

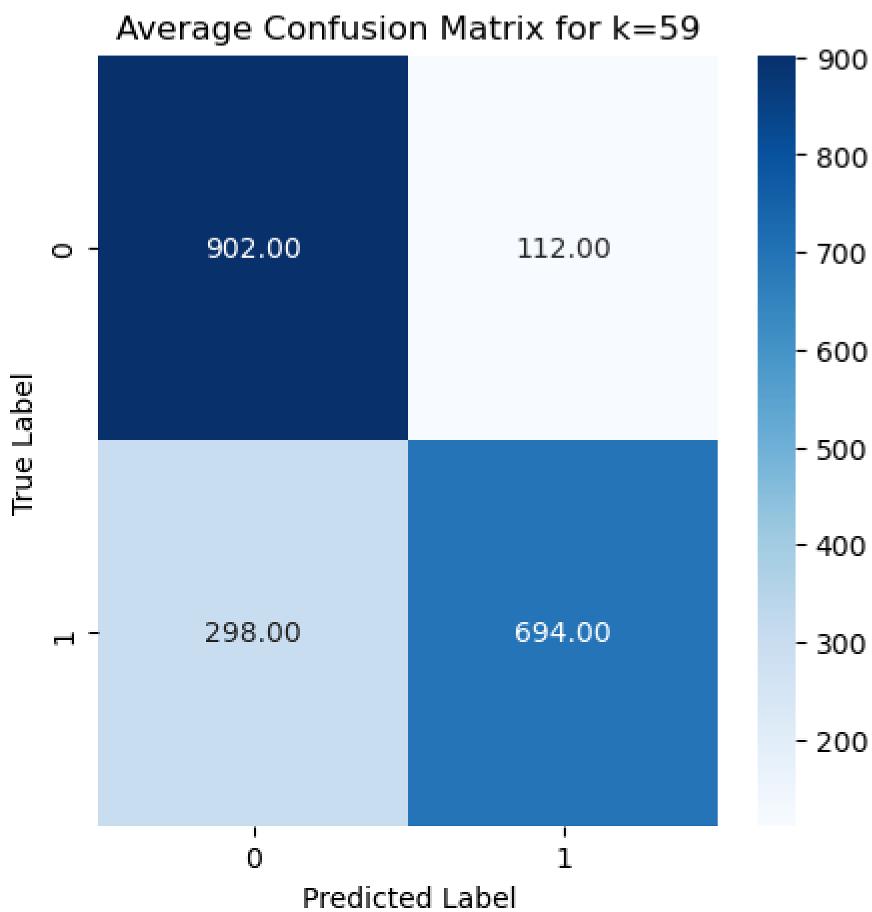

Average Confusion Matrix for k=49
902.00
112.00
294.00
698.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

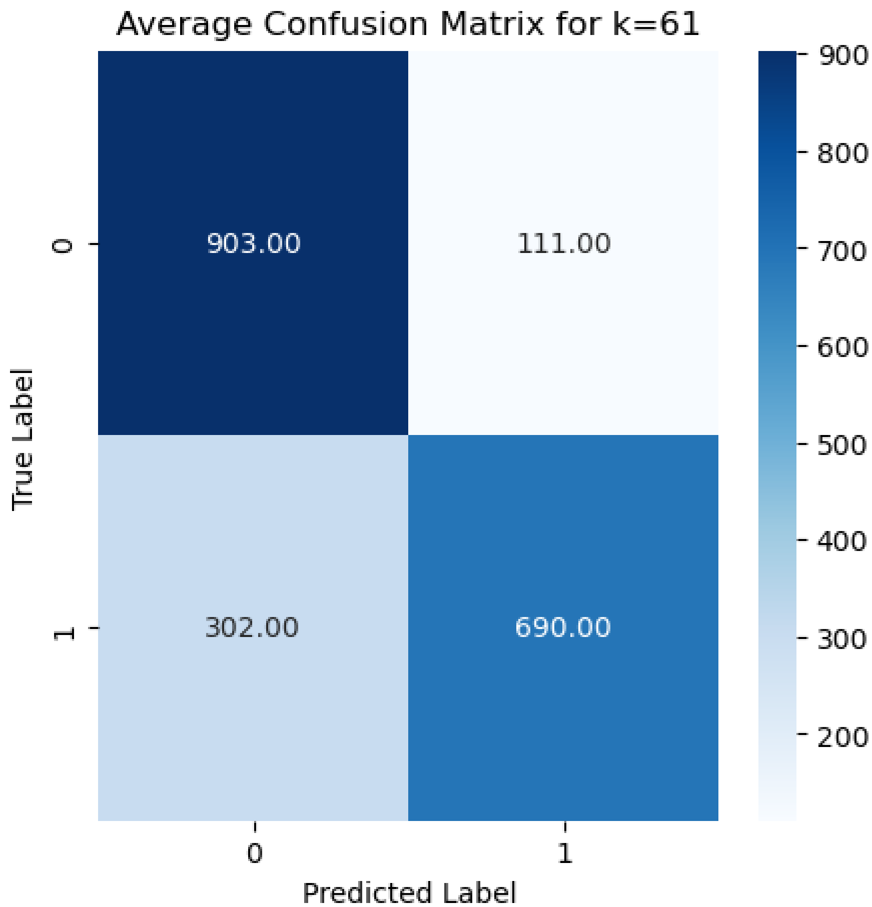

Average Confusion Matrix for k=51
903.00
111.00
294.00
698.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

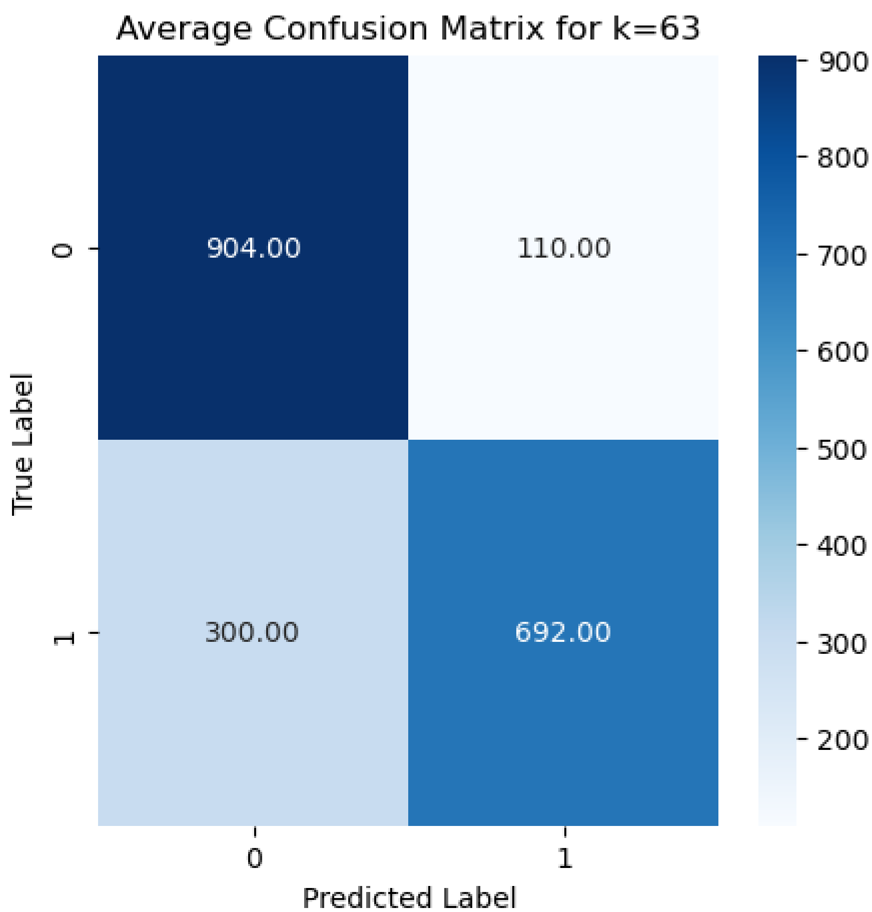

Average Confusion Matrix for k=53
True Label
Predicted Label
0
1
900.00
114.00
292.00
700.00
900
800
700
600
500
400
300
200

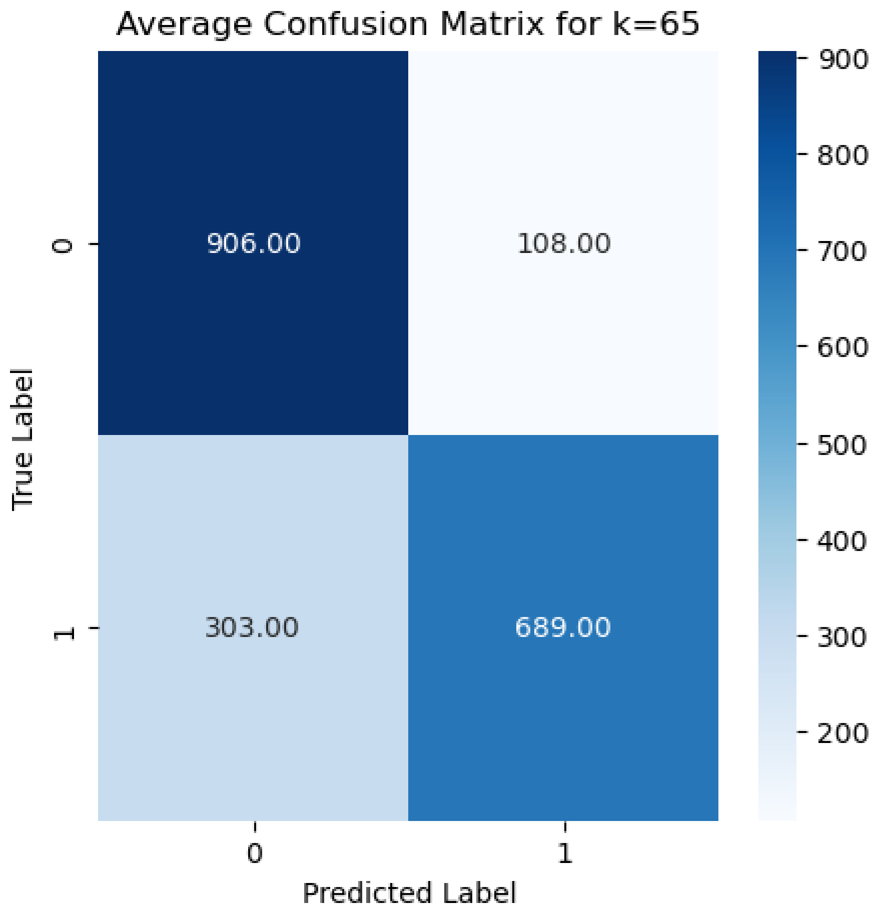

Average Confusion Matrix for k=55
True Label
Predicted Label
0
1
903.00
111.00
294.00
698.00
900
800
700
600
500
400
300
200

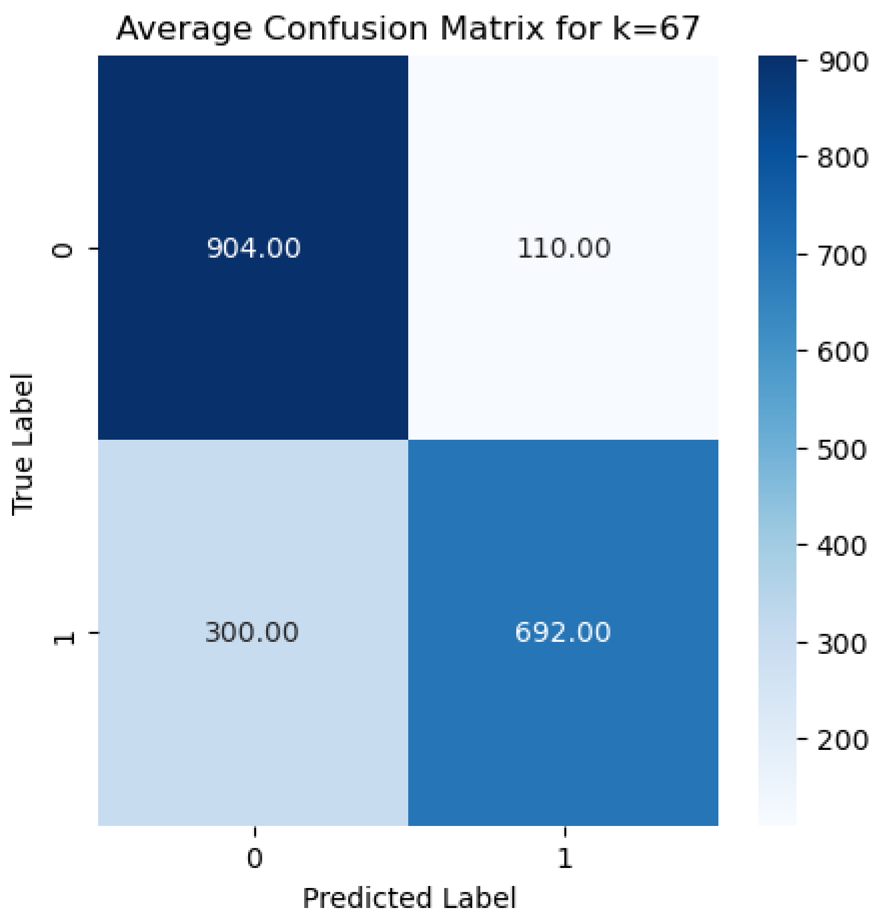

Average Confusion Matrix for k=57
True Label
Predicted Label
0
1
900.00
114.00
296.00
696.00
900
800
700
600
500
400
300
200

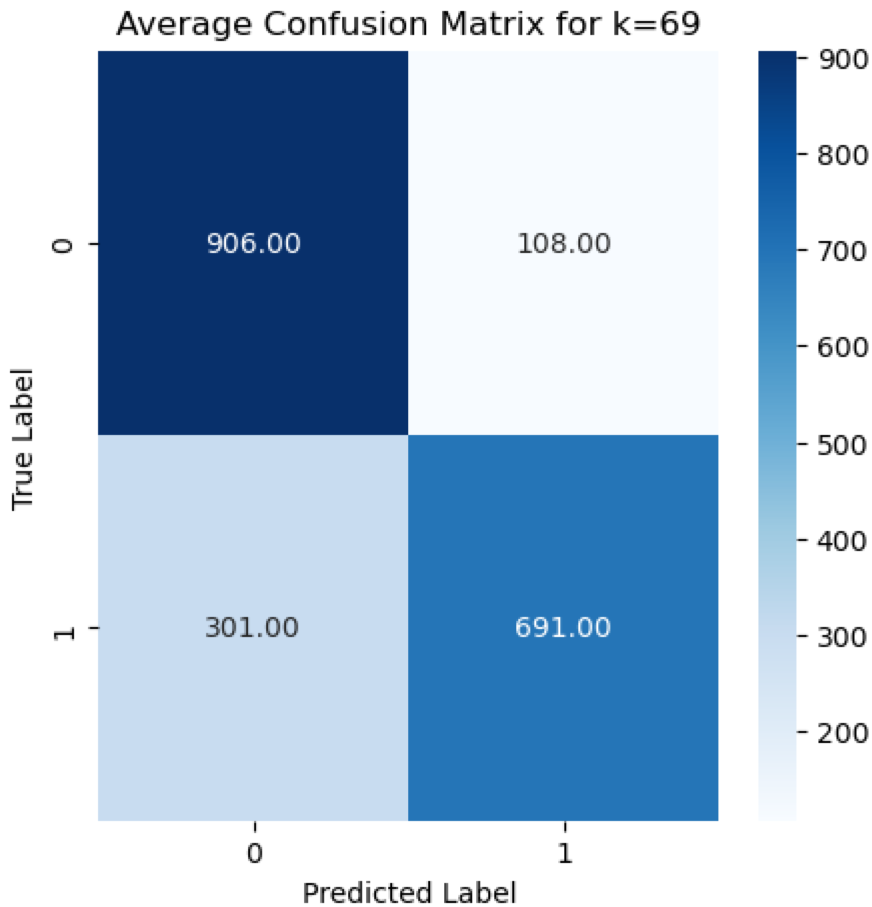

Average Confusion Matrix for k=59
True Label
Predicted Label
0
1
902.00
112.00
298.00
694.00
900
800
700
600
500
400
300
200

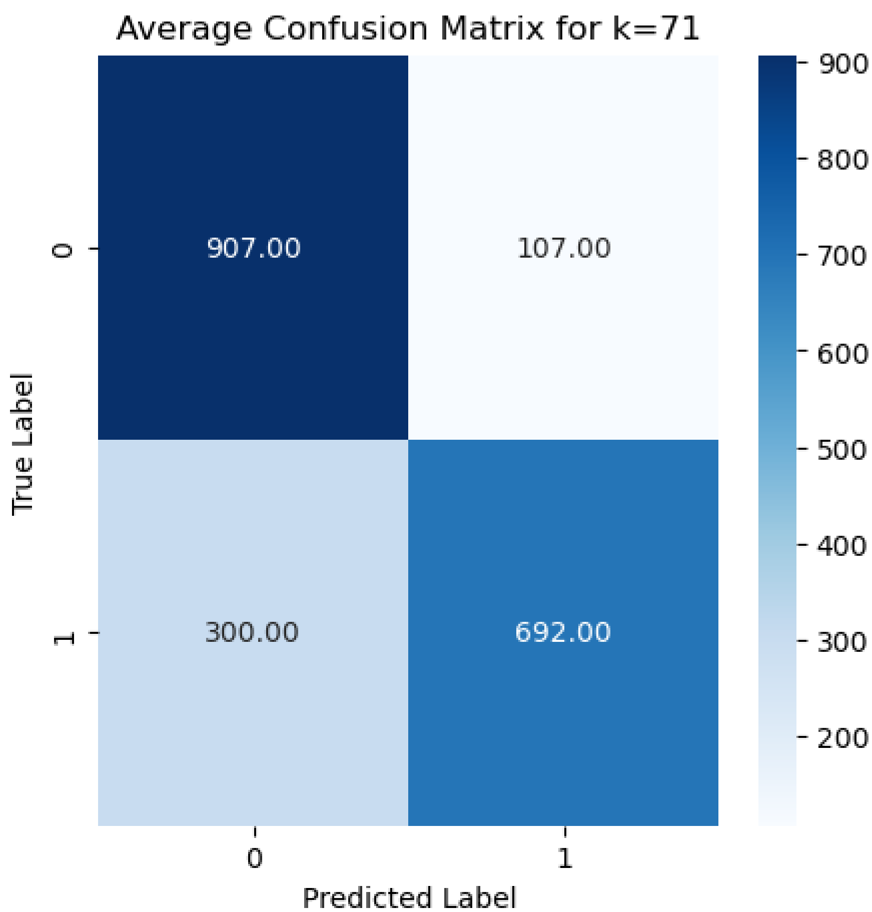

Average Confusion Matrix for k=61
903.00
111.00
302.00
690.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

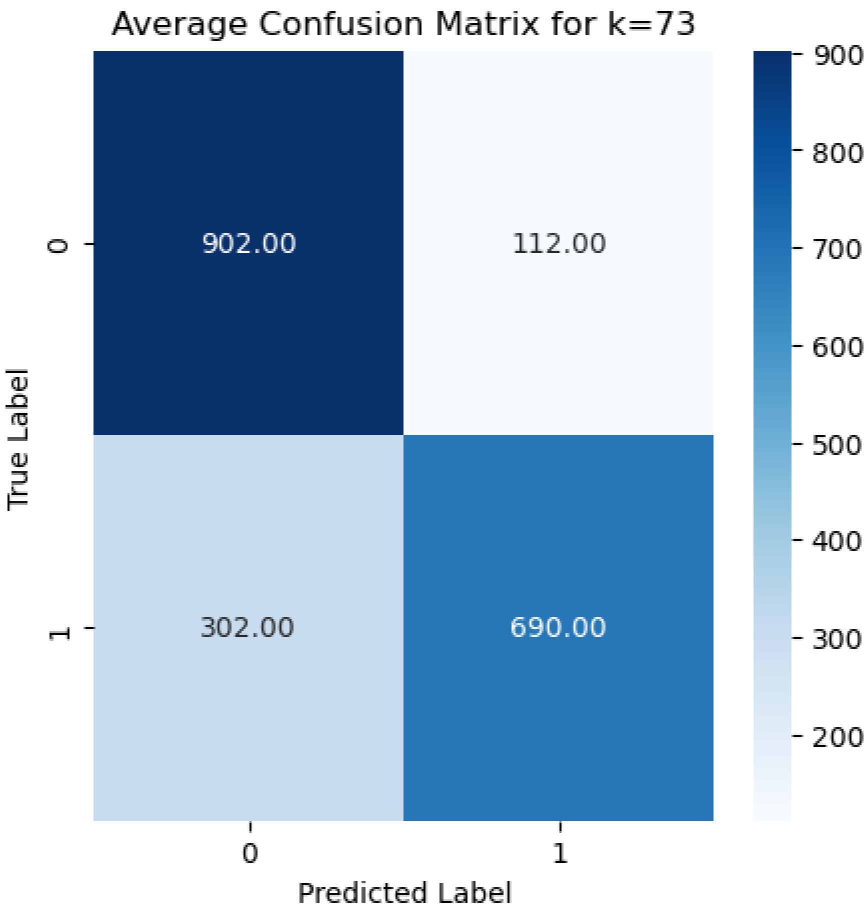

Average Confusion Matrix for k=63
904.00
110.00
300.00
692.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

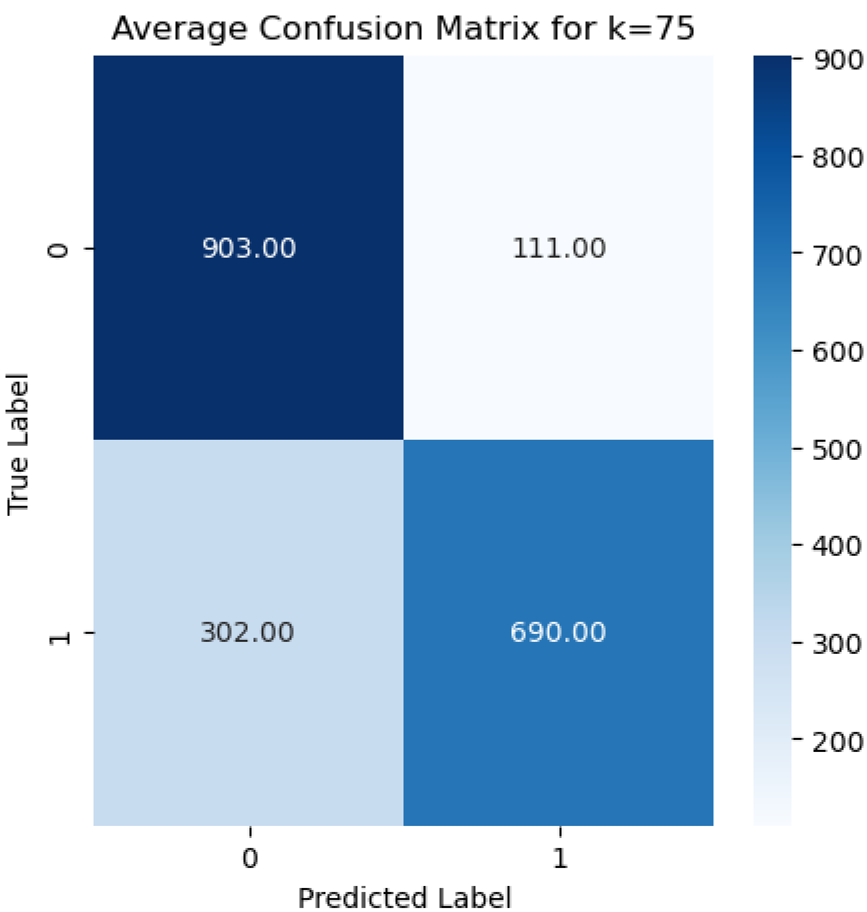

Average Confusion Matrix for k=65
True Label
Predicted Label
0
1
0
1
906.00
108.00
303.00
689.00
900
800
700
600
500
400
300
200

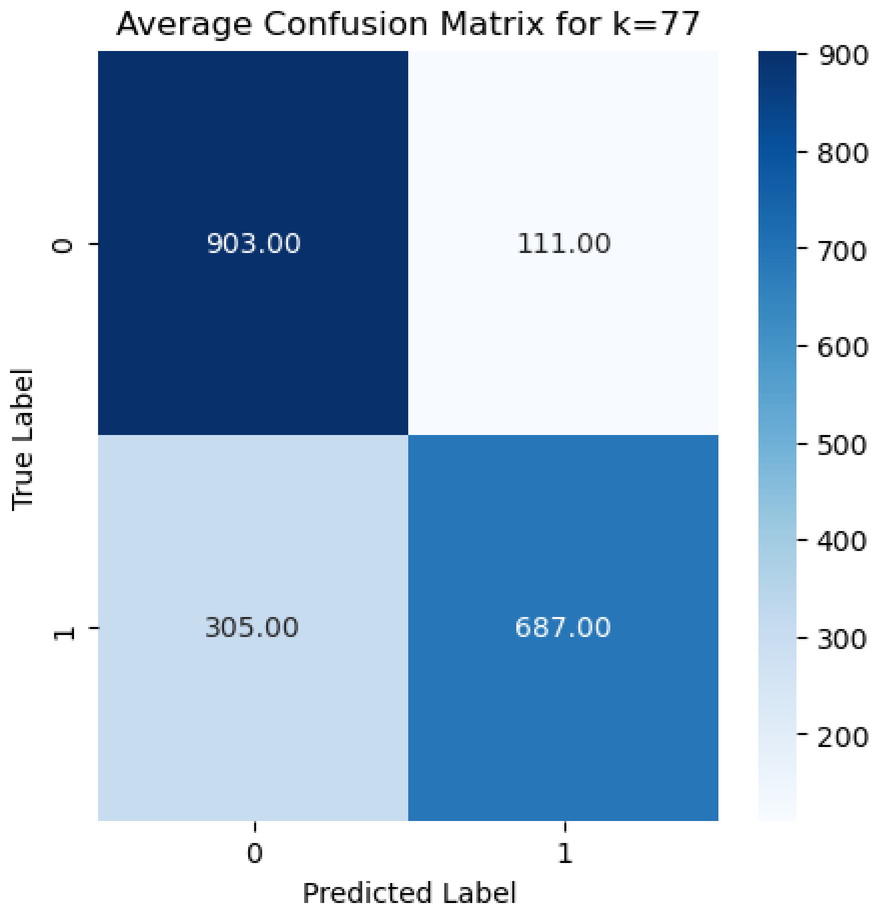

Average Confusion Matrix for k=67
True Label
Predicted Label
0
1
0
1
904.00
110.00
300.00
692.00
900
800
700
600
500
400
300
200

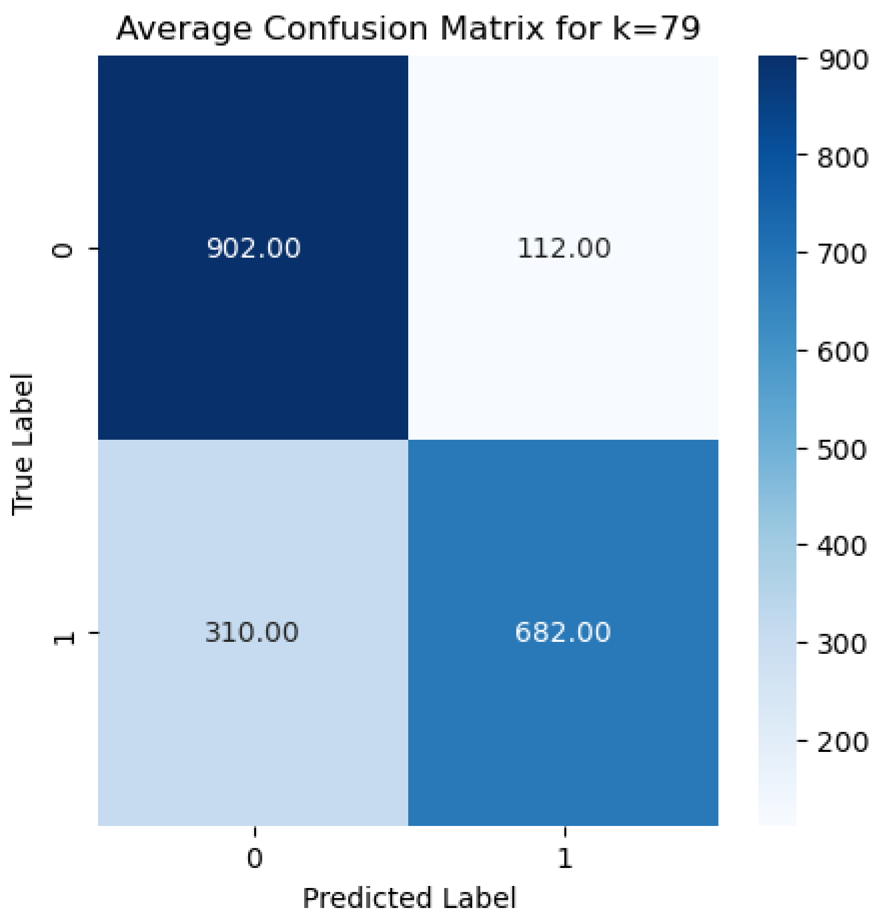

Average Confusion Matrix for k=69
906.00
108.00
301.00
691.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

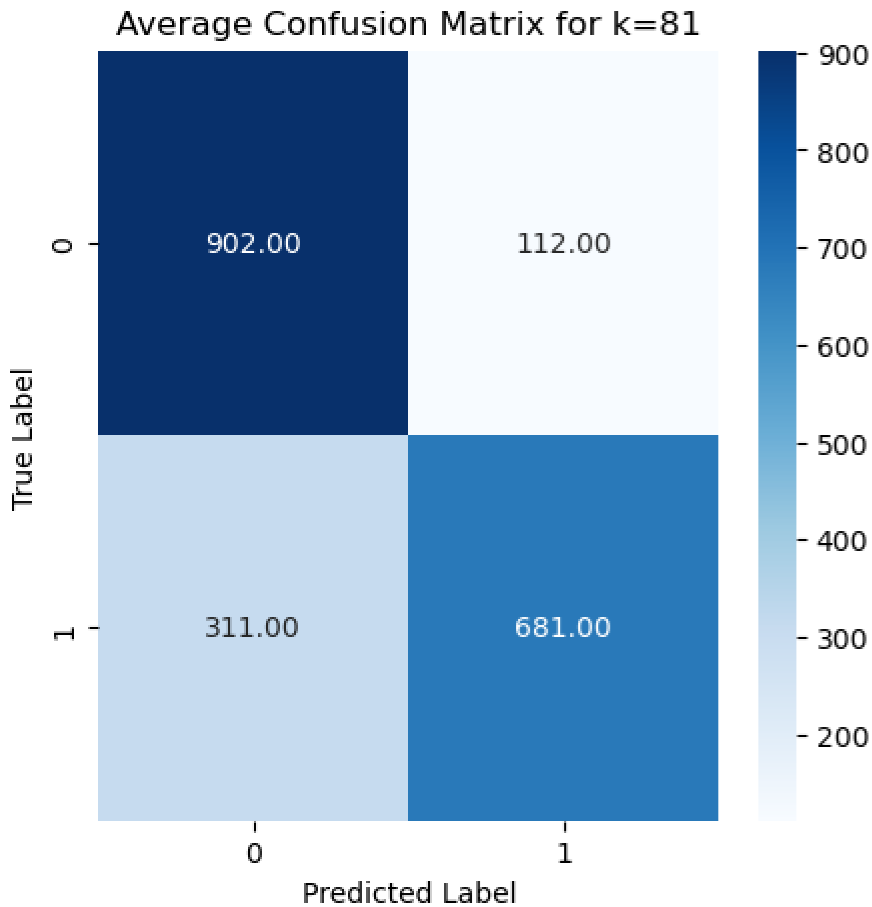

Average Confusion Matrix for k=71
907.00
107.00
300.00
692.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

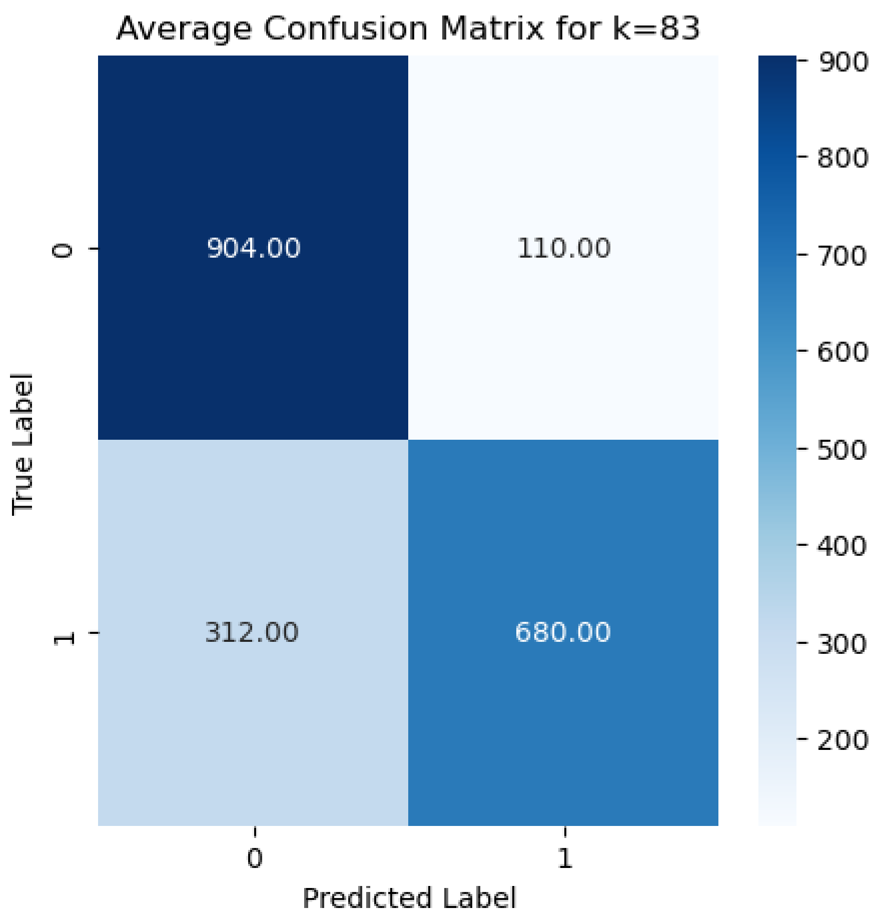

Average Confusion Matrix for k=73
902.00
112.00
302.00
690.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

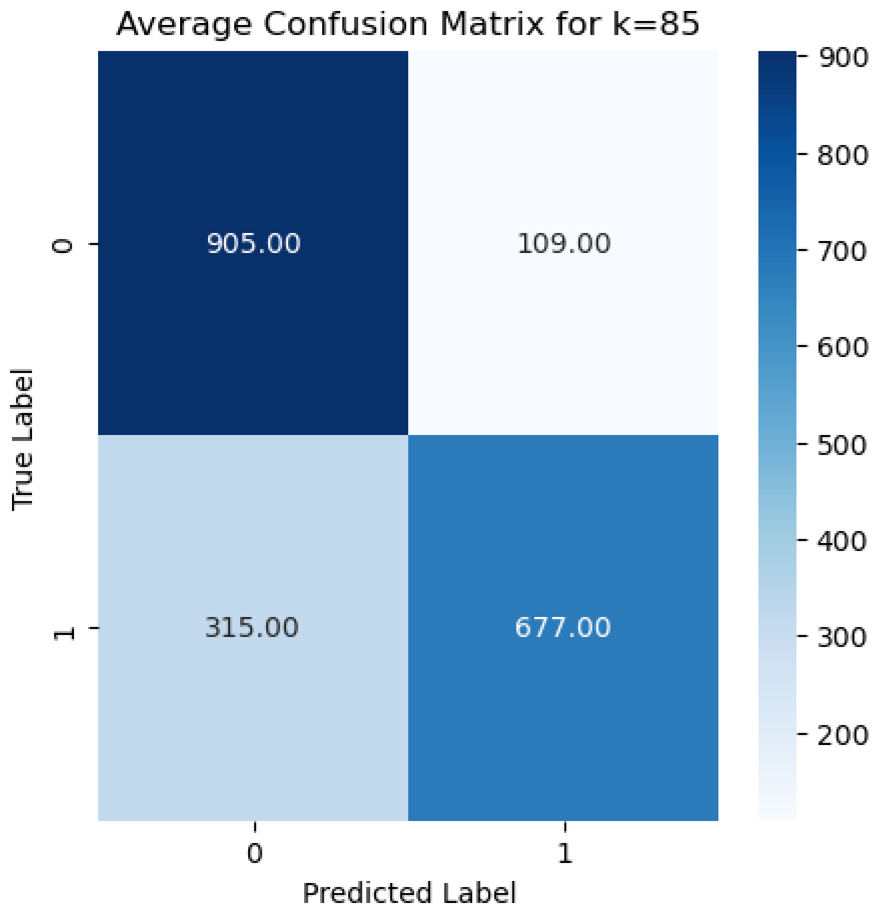

Average Confusion Matrix for k=75
903.00
111.00
302.00
690.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

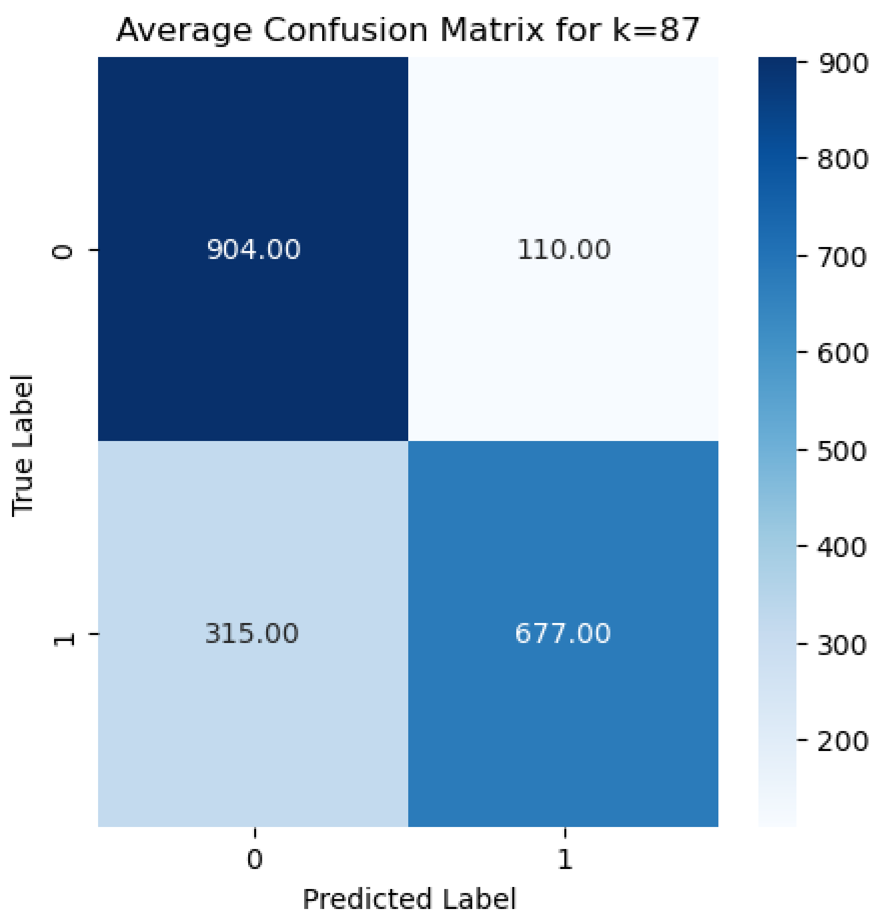

Average Confusion Matrix for k=77
903.00
111.00
305.00
687.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

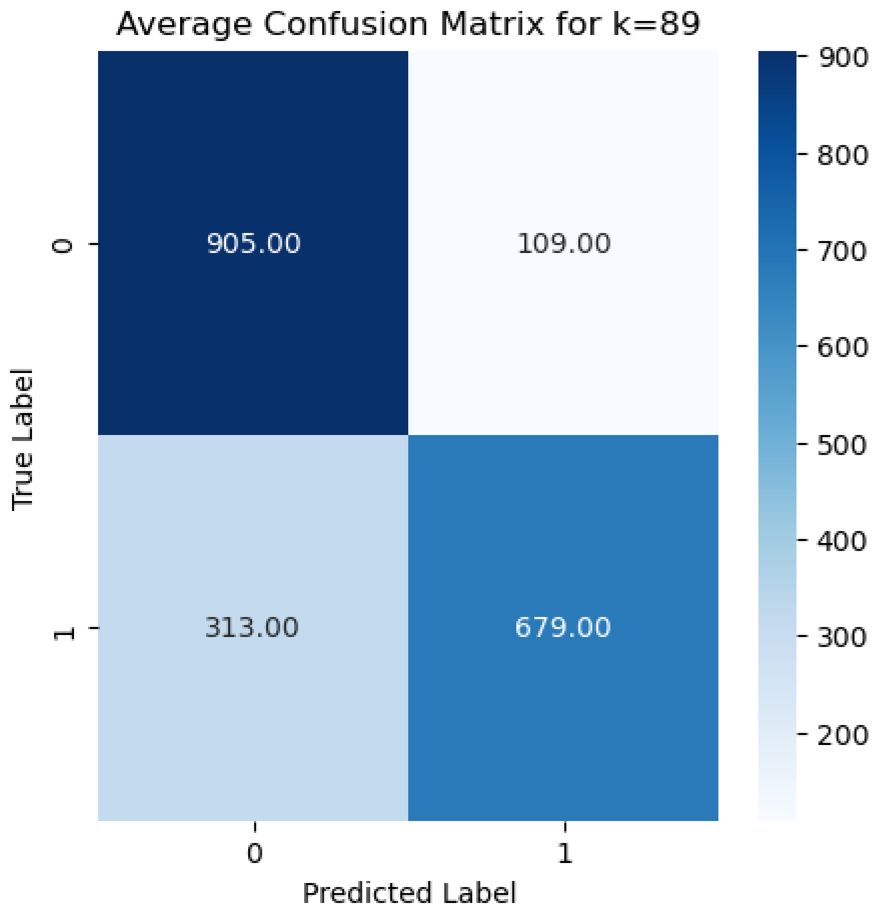

Average Confusion Matrix for k=79
902.00
112.00
310.00
682.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

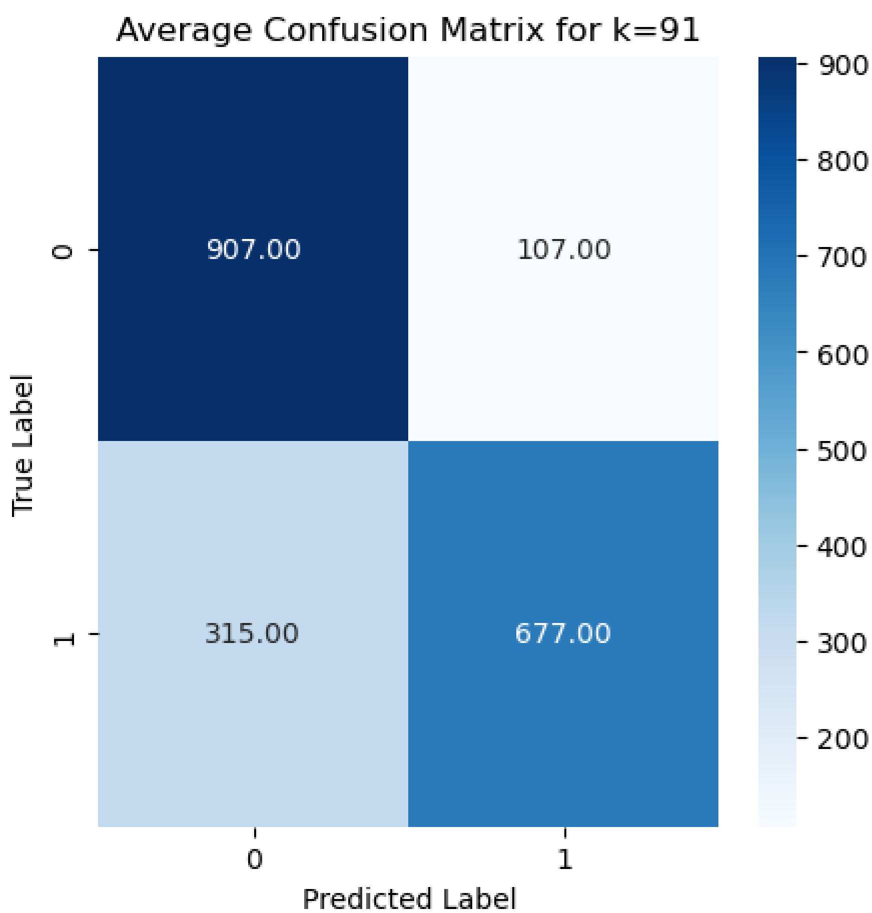

Average Confusion Matrix for k=81
902.00
112.00
311.00
681.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

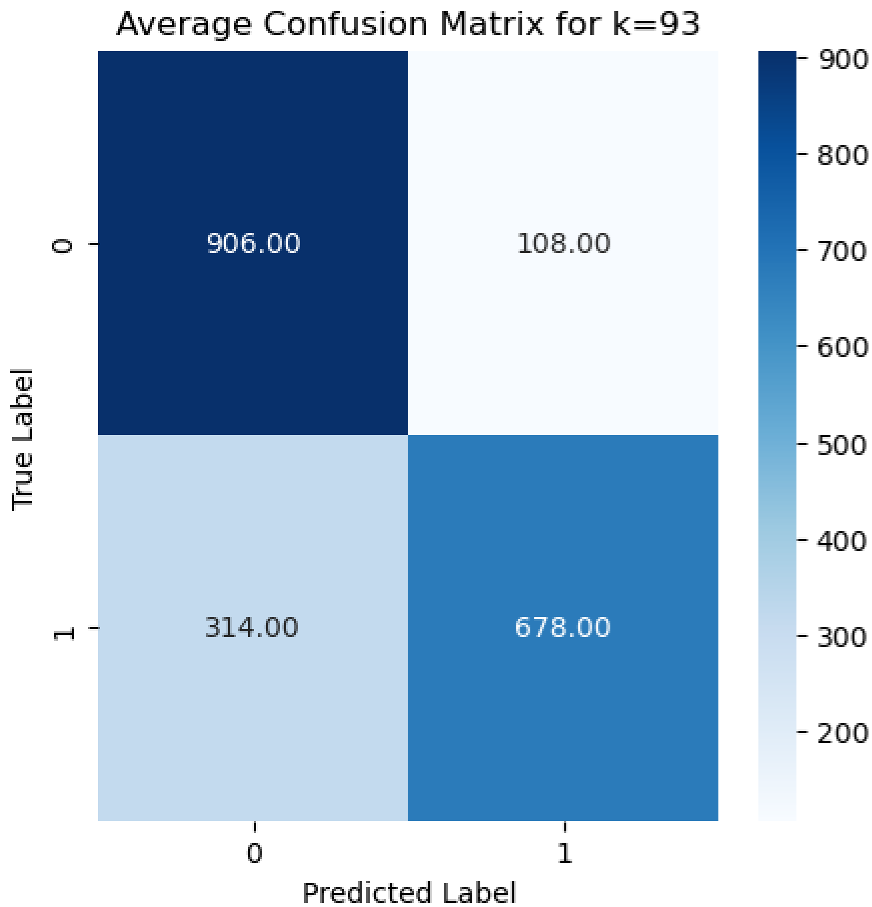

Average Confusion Matrix for k=83
904.00
110.00
312.00
680.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

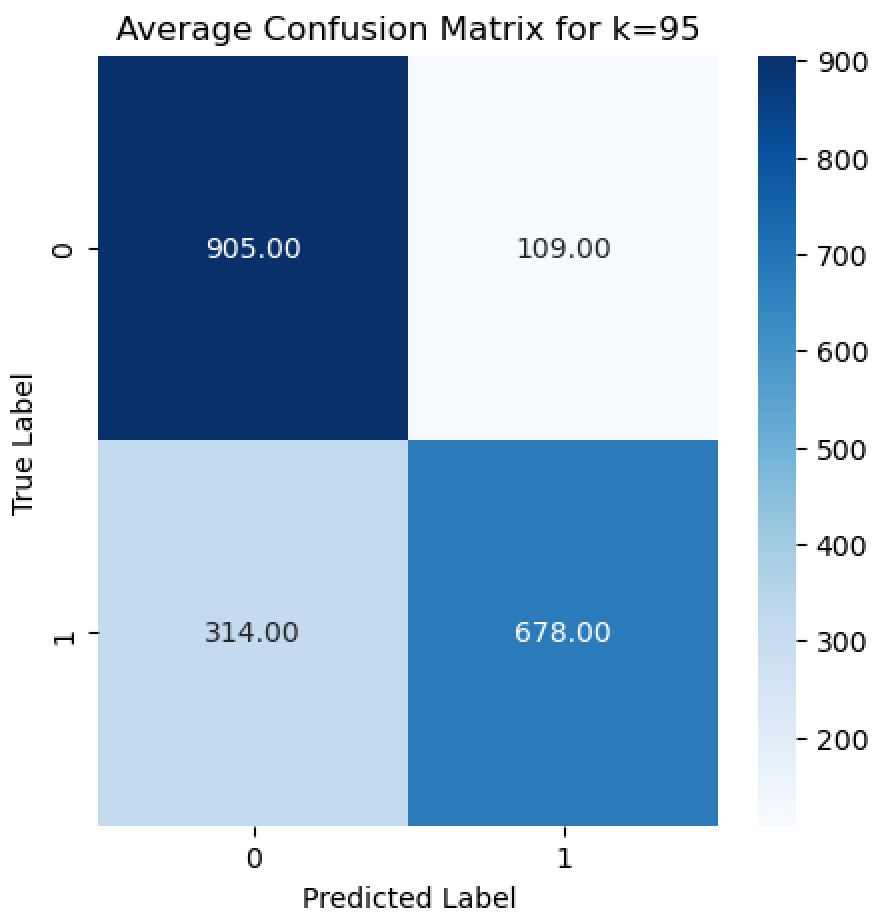

Average Confusion Matrix for k=85
905.00
109.00
315.00
677.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

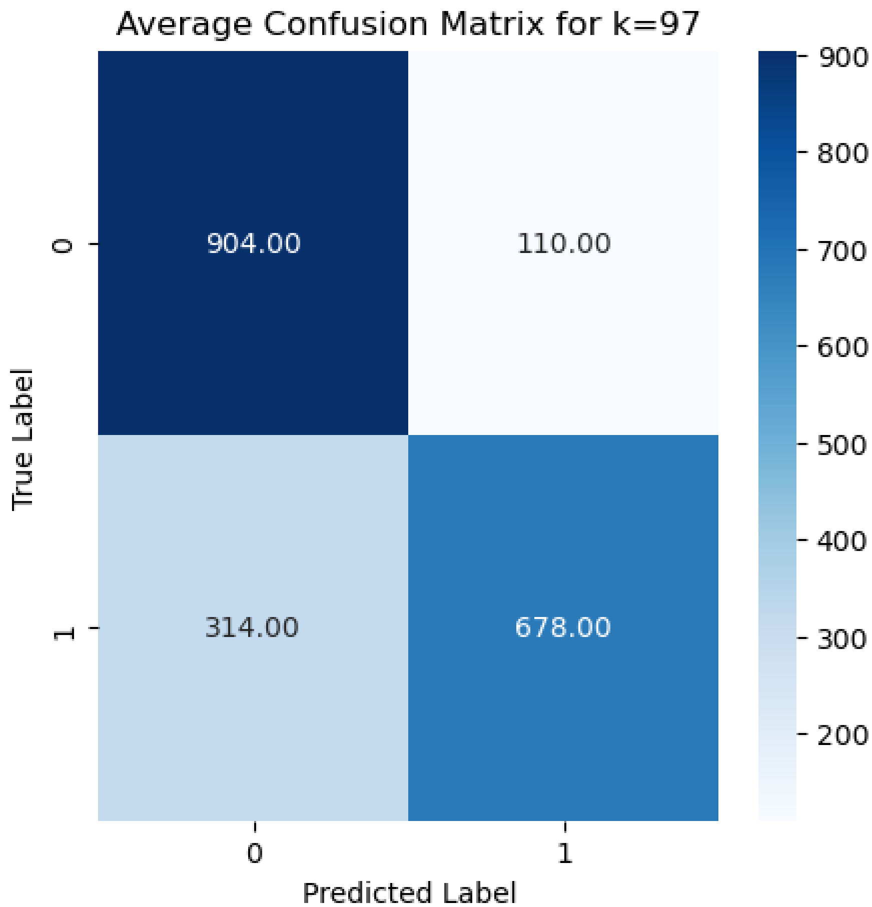

Average Confusion Matrix for k=87
904.00
110.00
315.00
677.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

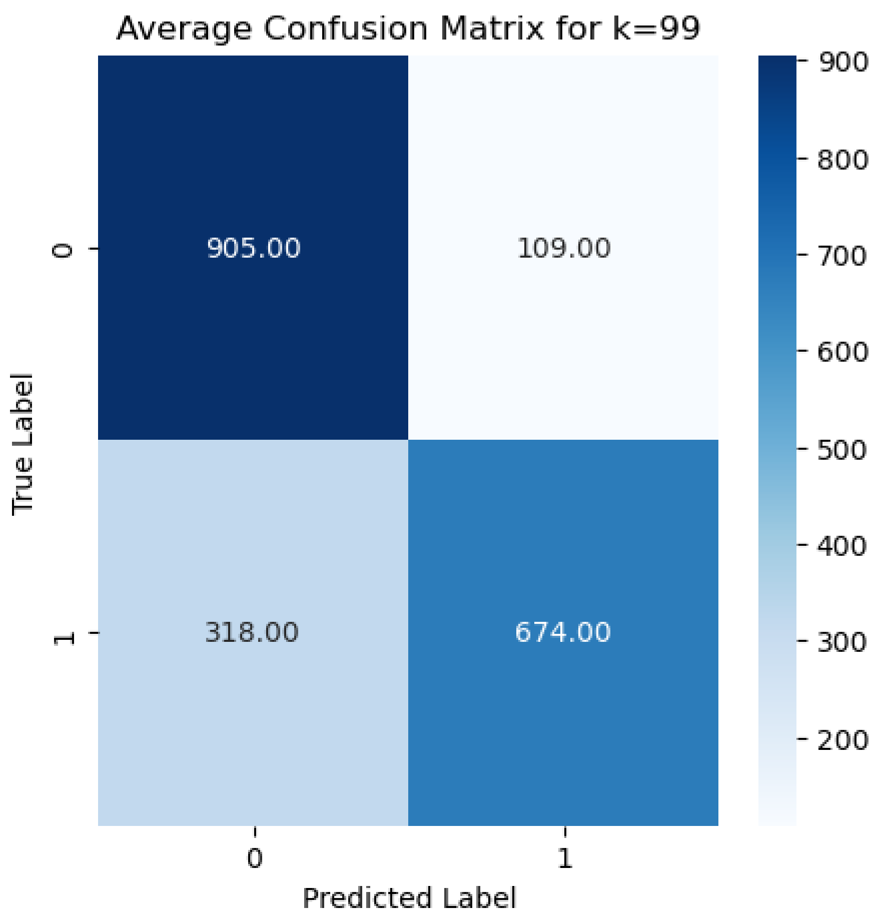

Average Confusion Matrix for k=89
905.00
109.00
313.00
679.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200


Average Confusion Matrix for k=91
907.00
107.00
315.00
677.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200


Average Confusion Matrix for k=93
906.00
108.00
314.00
678.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200


Average Confusion Matrix for k=95
905.00
109.00
314.00
678.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200


Average Confusion Matrix for k=97
904.00
110.00
314.00
678.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200


Average Confusion Matrix for k=99
905.00
109.00
318.00
674.00
True Label
Predicted Label
0
1
900
800
700
600
500
400
300
200

In [31]:
```python
best_k = neighbors[np.argmax(val_accs)]

print(f"best K value = {best_k}")
print(f"accuracy = {val_accs[np.argmax(val_accs)]}")
print(f"precision = {val_prec[np.argmax(val_accs)]}")
print(f"Recall = {val_rec[np.argmax(val_accs)]}")
print(f"F1 = {val_f1[np.argmax(val_accs)]}")
```

```
best K value = 9
accuracy = 0.8155533399800599
precision = 0.8624708624708625
Recall = 0.7459677419354839
F1 = 0.8
```

Testing

In [32]:
```python
model = KNeighborsClassifier(n_neighbors=best_k)
model.fit(X_train,y_train)
```

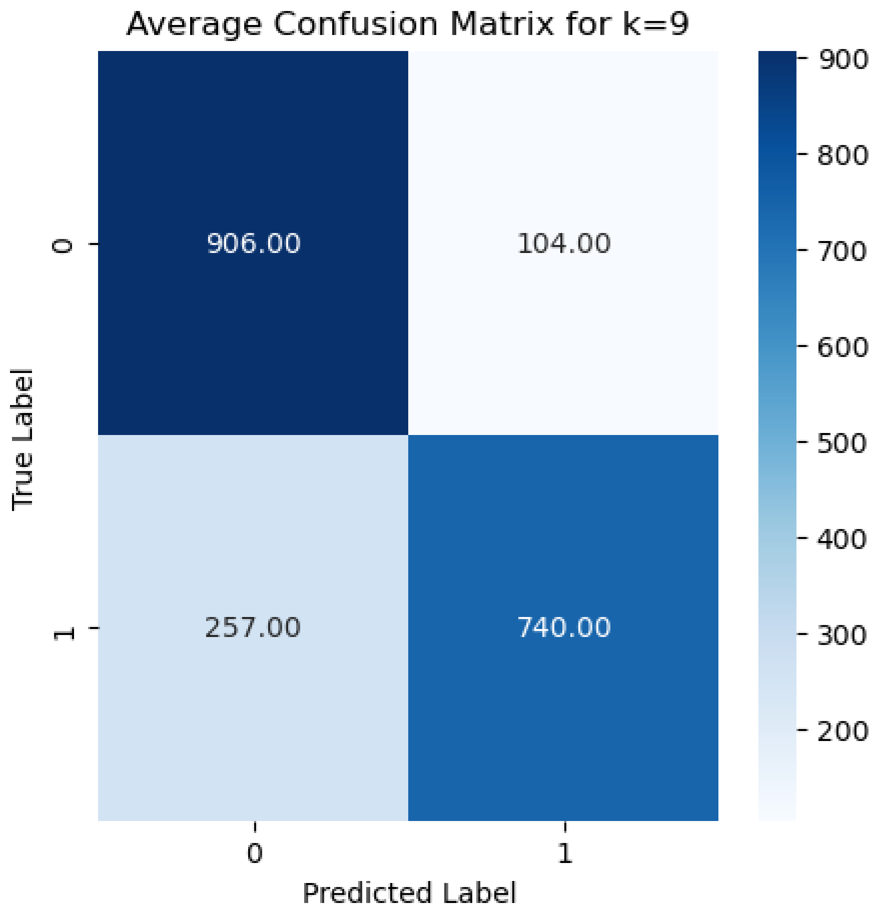
Out[32]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

In [33]:
```python
y_pred = model.predict(X_test)
```

In [34]:
```python
acc = accuracy_score(y_test,y_pred)
prec = precision_score(y_test,y_pred)
rec = recall_score(y_test,y_pred)
f1 = f1_score(y_test,y_pred)
print(f"accuracy = {round(acc,2)}")
print(f"precision = {round(prec,2)}")
print(f"Recall = {round(rec,2)}")
print(f"F1 = {round(f1,2)}")
```

```
accuracy = 0.82
precision = 0.88
Recall = 0.74
F1 = 0.8
```

In [35]:
```python
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 5))
sns.heatmap(cm, annot=True, fmt='.2f', cmap='Blues')
plt.title(f'Average Confusion Matrix for k={best_k}')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```


Average Confusion Matrix for k=9
906.00
104.00
257.00
740.00
True Label
Predicted Label
0
1
0
1
900
800
700
600
500
400
300
200

```
In [ ]:
```